MySQL Reference Manual

Copyright © 1997-2003 MySQL AB

Sumário

1 Informações Gerais 1

1.1 Sobre Este Manual 2
1.1.1 Convenções Usadas Neste Manual 2
1.2 Visão Geral do Sistema de Gerenciamento de Banco de Dados MySQL 4
1.2.1 História do MySQL 5
1.2.2 As Principais Características do MySQL 6
1.2.3 Estabilidade do MySQL 8
1.2.4 Qual o Tamanho Que as Tabelas do MySQL Podem Ter? 10
1.2.5 Compatibilidade Com o Ano 2000 (Y2K) 11
1.3 Visão Geral da MySQL AB 12
1.3.1 O Modelo de Negócio e Serviços da MySQL AB 13
1.3.1.1 Suporte 13
1.3.1.2 Treinamento e Certificação 14
1.3.1.3 Consultoria 14
1.3.1.4 Licenças Comerciais 15
1.3.1.5 Parcerias 15
1.3.2 Informações para Contato 15
1.4 Suporte e Licenciamento do MySQL 17
1.4.1 Suporte Oferecido pela MySQL AB 17
1.4.2 Copyrights e Licenças Usadas pelo MySQL 17
1.4.3 Licenças do MySQL 18
1.4.3.1 Usando o Programa MySQL Sob uma Licença Comercial 18
1.4.3.2 Usando o Programa MySQL Sem Custo Sob GPL 19
1.4.4 Logomarcas e Marcas Registradas da MySQL AB 20
1.4.4.1 O Logo Original do MySQL 20
1.4.4.2 Logomarcas da MySQL que Podem Ser Usadas Sem Permissão de Alteração 20
1.4.4.3 Quando Você Precisa de Permissão de Alteração para Usar as Logomarcas do MySQL? 21
1.4.4.4 Logomarcas dos Parceiros da MySQL AB 21
1.4.4.5 Usando a Palavra `MySQL` em Texto Impresso ou Apresentação 21
1.4.4.6 Usando a Palavra `MySQL` em Nomes de Companhias e Produtos 21
1.5 Mapa de Desenvolvimento do MySQL 22
1.5.1 MySQL 4.0 in a Nutshell 22

1.5.1.1 Recursos Disponíveis no MySQL 4.0 22
1.5.1.2 Servidor Embutido MySQL 24
1.5.2 MySQL 4.1 in a Nutshell 24
1.5.2.1 Recursos Disponíveis no MySQL 4.1 24
1.5.2.2 Stepwise Rollout 26
1.5.2.3 Pronto para Uso em Desenvolvimento Imediato 26
1.5.3 MySQL 5.0, A Próxima Distribuição de Desenvolvimento 26
1.6 MySQL e o Futuro (o TODO) 26
1.6.1 Novos Recursos Planejados Para a Versão 4.1 27
1.6.2 Novos Recursos Planejados Para a Versão 5.0 27
1.6.3 Novos Recursos Planejados Para a Versão 5.1 28
1.6.4 Novos Recursos Planejados Para a Versão em um Futuro Próximo 29
1.6.5 Novos Recursos Planejados Para a Versão em um Futuro a Médio Prazo 31
1.6.6 Novos Recursos que Não Planejamos Fazer 33
1.7 Fontes de Informações do MySQL 33
1.7.1 Listas de Discussão MySQL 33
1.7.1.1 As Listas de Discussão do MySQL 33
1.7.1.2 Fazendo perguntas ou relatando erros ... 35
1.7.1.3 Como relatar erros ou problemas 36
1.7.1.4 Guia para responder questões na lista de discussão 41
1.7.2 Suporte a Comunidade MySQL Atrvés do IRC (Internet Relay Chat) 41
1.8 Qual compatibilidade aos padrões o MySQL oferece ? 41
1.8.1 Qual Padrão o MySQL Segue? 42
1.8.2 Executando o MySQL no modo ANSI 42
1.8.3 Extensões do MySQL para o Padrão SQL-92 43
1.8.4 Diferenças do MySQL em Comparação com o SQL-92 46
1.8.4.1 Subqueries 46
1.8.4.2 `SELECT INTO TABLE` 46
1.8.4.3 Transações e Operações Atômicas 46
1.8.4.4 Stored Procedures e Triggers 49
1.8.4.5 Chaves Estrangeiras 50
1.8.4.6 Views 51
1.8.4.7 '`--`' como Início de Comentário 51
1.8.5 Como o MySQL Lida com Restrições 52
1.8.5.1 Restrições de PRIMARY KEY / UNIQUE 52
1.8.5.2 Restrições de `NOT NULL` 53
1.8.5.3 Restrições de `ENUM` e `SET` 53
1.8.6 Erros Conhecidos e Deficiências de Projetos no MySQL 54

1.8.6.1 Erros da Versão 3.23 Corrigidos em Versões Posteriores do MySQL 54
1.8.6.2 Open Bugs / Deficiências de Projeto no MySQL 54

2 Instalação do MySQL 60

2.1 Instalação rápida padrão do MySQL 60
2.1.1 Instalando o MySQL no Windows 60
2.1.1.1 Exigências do Sistema Windows 61
2.1.1.2 Instalando uma Distribuição Binária do Windows 61
2.1.1.3 Preparando o Ambiente MySQL do Windows 62
2.1.1.4 Selecionando um Servidor Windows 63
2.1.1.5 Iniciando o Servidor pela Primeira Vez 64
2.1.1.6 Iniciando o MySQL no Windows 95, 98, ou Me 65
2.1.1.7 Iniciando o MySQL no Windows NT, 2000, ou XP 65
2.1.1.8 Executando o MySQL no Windows 68
2.1.2 Instalando o MySQL no Linux 69
2.1.3 Instalando o MySQL no Mac OS X 71
2.1.4 Instalando o MySQL no NetWare 74
2.1.4.1 Instalando o MySQL para Binários do NetWare 74
2.2 Detalhes Gerais de Instalação 75
2.2.1 Como obter o MySQL 75
2.2.2 Verificando a Integridade do Pacote Usando `MD5 Checksums` ou `GnuPG` 75
2.2.3 Sistemas Operacionais suportados pelo MySQL 78
2.2.4 Qual versão do MySQL deve ser usada 80
2.2.5 Layouts de Instalação 83
2.2.6 Como e quando as atualizações são lançadas? 84
2.2.7 Filosofia das Distribuições - Nenhum Bug Conhecidos nas Distribuições 85
2.2.8 Binários MySQL compilados pela MySQL AB 86
2.2.9 Instalando uma Distribuição Binária do MySQL 91
2.3 Instalando uma distribuição com fontes do MySQL 94
2.3.1 Visão geral da instalação rápida 95
2.3.2 Aplicando patches 97
2.3.3 Opções típicas do `configure` 97
2.3.4 Instalando pela árvore de fontes do desenvolvimento 100
2.3.5 Lidando com Problemas de Compilação 103
2.3.6 Notas MIT-pthreads 106
2.3.7 Instalando o MySQL a partir do Fonte no Windows 107
2.3.7.1 Construindo o MySQL Usando VC++ 108

2.3.7.2 Criando um Pacote Fonte do Windows a partir da Última Fonte de Desenvolvimento 110
2.4 Configurações e Testes Pós-instalação 111
2.4.1 Problemas Executando o `mysql_install_db` 115
2.4.2 Problemas Inicializando o Servidor MySQL 116
2.4.3 Inicializando e parando o MySQL automaticamente. 118
2.5 Atualizando/Desatualizando o MySQL 120
2.5.1 Atualizando da Versão 4.0 para 4.1 120
2.5.2 Atualizando da Versão 3.23 para 4.0 123
2.5.3 Atualizando da versão 3.22 para 3.23 127
2.5.4 Atualizando da versão 3.21 para 3.22 128
2.5.5 Atualizando da versão 3.20 para 3.21 129
2.5.6 Atualizando a Tabela de Permissões 130
2.5.7 Atualizando para outra arquitetura 131
2.5.8 Atualizando o MySQL no Windows 132
2.6 Notas específicas para os Sistemas Operacionais 133
2.6.1 Notas Windows 133
2.6.1.1 Conectando em um MySQL Rematamente a Windows Utilizando SSH 133
2.6.1.2 Compilando clientes MySQL no Windows 133
2.6.1.3 MySQL para Windows Comparado com o MySQL para Unix 134
2.6.2 Notas Linux (Todas as versões) 136
2.6.2.1 Notas Linux para distribuições binárias 140
2.6.2.2 Notas Linux x86 142
2.6.2.3 Notas Linux SPARC 143
2.6.2.4 Notas Linux Alpha 143
2.6.2.5 Notas Linux PowerPC 144
2.6.2.6 Notas Linux MIPS 144
2.6.2.7 Notas Linux IA-64 144
2.6.3 Notas Solaris 144
2.6.3.1 Notas Solaris 2.7/2.8 147
2.6.3.2 Notas Solaris x86 148
2.6.4 Notas BSD 148
2.6.4.1 Notas FreeBSD 148
2.6.4.2 Notas NetBSD 150
2.6.4.3 Notas OpenBSD 150
2.6.4.4 Notas OpenBSD 2.8 150
2.6.4.5 Notas BSDI Versão 2.x 150
2.6.4.6 Notas BSD/OS Versão 3.x 151
2.6.4.7 Notas BSD/OS Versão 4.x 151
2.6.5 Notas Mac OS X 152
2.6.5.1 Mac OS X 10.x 152
2.6.5.2 Mac OS X Server 1.2 (Rhapsody) 152

2.6.6 Notas de Outros Unix 152
2.6.6.1 Notas HP-UX para distribuições binárias 153
2.6.6.2 Notas HP-UX Versão 10.20 153
2.6.6.3 Notas HP-UX Versão 11.x 154
2.6.6.4 Notas IBM-AIX 155
2.6.6.5 Notas SunOS 4 157
2.6.6.6 Notas Alpha-DEC-UNIX (Tru64) 157
2.6.6.7 Notas Alpha-DEC-OSF1 158
2.6.6.8 Notas SGI Irix 159
2.6.6.9 Notas SCO 160
2.6.6.10 Notas SCO Unixware Version 7.0 163
2.6.7 Notas OS/2 163
2.6.8 Notas Novell NetWare 164
2.6.9 Notas BeOS 164
2.7 Comentários de Instalação do Perl 164
2.7.1 Instalando Perl no Unix 164
2.7.2 Instalaando ActiveState Perl no Windows 165
2.7.3 Problemas Usando a Interface Perl `DBI/DBD` 166

3 Tutorial de Introdução Do MySQL 168

3.1 Conectando e Desconectando do Servidor 168
3.2 Fazendo Consultas 169
3.3 Criação e Utilização de um Banco de Dados 172
3.3.1 Criando e Selecionando um Banco de Dados 173
3.3.2 Criando uma Tabela 174
3.3.3 Carregando dados em uma tabela 175
3.3.4 Recuperando Informações de uma Tabela 177
3.3.4.1 Selecionando Todos os Dados 177
3.3.4.2 Selecionando Registros Específicos 178
3.3.4.3 Selecionando Colunas Específicas 179
3.3.4.4 Ordenando Registros 180
3.3.4.5 Cálculo de Datas 182
3.3.4.6 Trabalhando com Valores Nulos (`NULL`) 185
3.3.4.7 Combinação de padrões 185
3.3.4.8 Contando Registros 188
3.3.4.9 Utilizando Múltiplas Tabelas 190
3.4 Obtendo Informações Sobre Bancos de Dados e Tabelas . . 192
3.5 Utilizando `mysql` em Modo Batch 193
3.6 Exemplos de Consultas Comuns 195
3.6.1 O Valor Máximo para uma Coluna 195
3.6.2 O Registro que Armazena o Valor Máximo para uma Coluna Determinada 196
3.6.3 Máximo da Coluna por Grupo 196
3.6.4 As Linhas Armazenando o Group-wise Máximo de um Certo Campo 197
3.6.5 Utilizando Variáveis de Usuário 198

3.6.6 Utilizando Chaves Estrangeiras 198
3.6.7 Pesquisando em Duas Chaves 200
3.6.8 Calculando Visitas Diárias 200
3.6.9 Usando `AUTO_INCREMENT` 201
3.7 Consultas de Projetos Gêmeos 202
3.7.1 Encontrando Todos Gêmeos Não-distribuídos . . . 203
3.7.2 Mostrando uma Tabela sobre a Situação dos Pares Gêmeos 205
3.8 Utilizando MySQL com Apache 206

4 Administração do Bancos de Dados MySQL 207

4.1 Configurando o MySQL 207
4.1.1 Opções de Linha de Comando do `mysqld` 207
4.1.2 Arquivo de Opções '`my.cnf`' 216
4.2 Executando Múltiplos MySQL Servers na Mesma Máquina 219
4.2.1 Executando Múltiplos Servidores no Windows . . 220
4.2.1.1 Iniciando Múltiplos Servidores na Linha de Comando 221
4.2.1.2 Iniciando Múltiplos Servidores Como Serviços 222
4.2.2 Executando Múltiplos Servidores no Unix 224
4.2.3 Usando Programas Clientes em um Ambiente Multi-Servidor 225
4.3 Detalhes Gerais de Segurança e o Sistema de Privilégio de Acesso do MySQL 226
4.3.1 Segurança Geral 226
4.3.2 Como Tornar o MySQL Seguro contra Crackers 229
4.3.3 Opções de Inicialização para o `mysqld` em Relação a Segurança. 230
4.3.4 Detalhes de Segurança com `LOAD DATA LOCAL` . . . 231
4.3.5 O Que o Sistema de Privilégios Faz 232
4.3.6 Como o Sistema de Privilégios Funciona 232
4.3.7 Privilégios Fornecidos pelo MySQL 236
4.3.8 Conectando ao Servidor MySQL 238
4.3.9 Controle de Acesso, Estágio 1: Verificação da Conexão 239
4.3.10 Controle de Acesso, Estágio 2: Verificação da Requisição 242
4.3.11 Hashing de Senhas no MySQL 4.1 245
4.3.12 Causas dos Erros de `Accesso Negado` 249
4.4 Gerenciamento das Contas dos Usuários no MySQL 254
4.4.1 A Sintaxe de `GRANT` e `REVOKE` 254
4.4.2 Nomes de Usuários e Senhas do MySQL 259
4.4.3 Quando as Alterações nos Privilégios tem Efeito 260

4.4.4 Configurando os Privilégios Iniciais do MySQL . . 260
4.4.5 Adicionando Novos Usuários ao MySQL 261
4.4.6 Deletando Usuários do MySQL 264
4.4.7 Limitando os Recursos dos Usuários 265
4.4.8 Configurando Senhas 266
4.4.9 Mantendo Sua Senha Segura 267
4.4.10 Usando Conexões Seguras 268
4.4.10.1 Conceitos Basicos 268
4.4.10.2 Exigências 268
4.4.10.3 Configurando Certificados SSL para o MySQL 269
4.4.10.4 Opções SSL do `GRANT` 273
4.4.10.5 Opções SSL de Linha de Comando . . . 274
4.5 Prevenção de Disastres e Recuperação 275
4.5.1 Backups dos Bancos de Dados 275
4.5.2 Sintaxe de `BACKUP TABLE` 277
4.5.3 Sintaxe de `RESTORE TABLE` 277
4.5.4 Sintaxe de `CHECK TABLE` 278
4.5.5 Sintaxe do `REPAIR TABLE` 279
4.5.6 Utilizando `myisamchk` para Manutenção de Tabelas e Recuperação em Caso de Falhas 280
4.5.6.1 Sintaxe do `myisamchk` 281
4.5.6.2 Opções Gerais do `myisamchk` 282
4.5.6.3 Opções de Verificação do `myisamchk` . . . 283
4.5.6.4 Opções de Reparos do `myisamchk` 284
4.5.6.5 Outras Opções do `myisamchk` 286
4.5.6.6 Uso de Memória do `myisamchk` 286
4.5.6.7 Uso do `myisamchk` para Recuperação em Caso de Falhas 287
4.5.6.8 Como Verificar Erros em Tabelas 288
4.5.6.9 Como Reparar Tabelas 289
4.5.6.10 Otimização de Tabelas 291
4.5.7 Configurando um Regime de Manutenção das Tabelas 291
4.5.8 Obtendo Informações sobre as Tabelas 292
4.6 Adiministração do Banco de Dados e Referência de Linguagem 298
4.6.1 Sintaxe de `OPTIMIZE TABLE` 298
4.6.2 Sintaxe de `ANALYZE TABLE` 298
4.6.3 Sintaxe de `CHECKSUM TABLE` 299
4.6.4 Sintaxe de `FLUSH` 299
4.6.5 Sintaxe de `RESET` 301
4.6.6 Sintaxe de `PURGE MASTER LOGS` 301
4.6.7 Sintaxe de `KILL` 301
4.6.8 Sintaxe de `SHOW` 302
4.6.8.1 Recuperando Informações sobre Bancos de Dados, Tabelas, Colunas e Índices 303
4.6.8.2 `SHOW TABLE STATUS` 304

4.6.8.3 SHOW STATUS 305
4.6.8.4 SHOW VARIABLES 308
4.6.8.5 SHOW [BDB] LOGS 320
4.6.8.6 SHOW PROCESSLIST 320
4.6.8.7 SHOW GRANTS 322
4.6.8.8 SHOW CREATE TABLE 322
4.6.8.9 SHOW WARNINGS | ERRORS 322
4.6.8.10 SHOW TABLE TYPES 324
4.6.8.11 SHOW PRIVILEGES 325
4.7 Localização do MySQL e Utilização Internacional 325
4.7.1 O Conjunto de Caracteres Utilizado para Dados e Ordenação 325
4.7.1.1 German character set 326
4.7.2 Mensagens de Erros em Outras Línguas 327
4.7.3 Adicionando um Novo Conjunto de Caracteres . . 327
4.7.4 Os Vetores de Definições de Caracteres 329
4.7.5 Suporte à Ordenação de Strings 329
4.7.6 Suporte à Caracteres Multi-byte 330
4.7.7 Problemas com Conjuntos de Caracteres 330
4.8 Utilitários e Scripts do Lado do Servidor MySQL 330
4.8.1 Visão Geral dos Scripts e Utilitários do Lado Servidor 330
4.8.2 mysqld-safe, o wrapper do mysqld 331
4.8.3 mysqld_multi, programa para gerenciar múltiplos servidores MySQL 333
4.8.4 myisampack, O Gerador de Tabelas Compactadas de Somente Leitura do MySQL 336
4.8.5 mysqld-max, om servidor mysqld extendido 343
4.9 Utilitários e Scripts do Lado do Cliente MySQL 345
4.9.1 Visão Geral dos Utilitários e Scripts do Lado do Cliente 345
4.9.2 mysql, A Ferramenta de Linha de Comando 346
4.9.3 mysqlcc, The MySQL Control Center 354
4.9.4 mysqladmin, Administrando um Servidor MySQL 356
4.9.5 mysqlbinlog, Executando as Consultas a Partir de um Log Binário 358
4.9.6 Usando mysqlcheck para Manutenção de Tabelas e Recuperação em Caso de Falhas 359
4.9.7 mysqldump, Descarregando a Estrutura de Tabelas e Dados 361
4.9.8 mysqlhotcopy, Copiando Bancos de Dados e Tabelas do MySQL 366
4.9.9 mysqlimport, Importando Dados de Arquivos Texto 367
4.9.10 mysqlshow, Exibindo Bancos de Dados, Tabelas e Colunas 369

4.9.11 `mysql_config`, Opções para compilação do cliente MySQL 370
4.9.12 `perror`, Explicando Códigos de Erros 371
4.9.13 Como Executar Comandos SQL a Partir de um Arquivo Texto 371
4.10 Os Arquivos de Log do MySQL 372
4.10.1 O Log de Erros 372
4.10.2 O Log de Consultas 373
4.10.3 O Log de Atualizações 373
4.10.4 O Log Binário 374
4.10.5 O Log para Consultas Lentas 377
4.10.6 Manutenção do Log de Arquivo 377
4.11 Replicação no MySQL 378
4.11.1 Introdução 378
4.11.2 Visão Geral da Implementação da Replicação 379
4.11.3 Detalhes de Implementação da Replicação 380
4.11.4 Como Configurar a Replicação 385
4.11.5 Recursos de Replicação e Problemas Conhecidos 389
4.11.6 Opções de Inicialização da Replicação 392
4.11.7 Instruções SQL para Controle do Servidor Master 400
4.11.7.1 `PURGE MASTER LOGS` 400
4.11.7.2 `RESET MASTER` 401
4.11.7.3 `SET SQL_LOG_BIN` 401
4.11.7.4 `SHOW BINLOG EVENTS` 401
4.11.7.5 `SHOW MASTER STATUS` 401
4.11.7.6 `SHOW MASTER LOGS` 401
4.11.7.7 `SHOW SLAVE HOSTS` 401
4.11.8 Instruções SQL para Controle do Servidor Slave 401
4.11.8.1 `CHANGE MASTER TO` 402
4.11.8.2 `LOAD DATA FROM MASTER` 404
4.11.8.3 `LOAD TABLE tbl_name FROM MASTER` ... 405
4.11.8.4 `MASTER_POS_WAIT()` 405
4.11.8.5 `RESET SLAVE` 405
4.11.8.6 `SET GLOBAL SQL_SLAVE_SKIP_COUNTER` 405
4.11.8.7 `SHOW SLAVE STATUS` 406
4.11.8.8 `START SLAVE` 409
4.11.8.9 `STOP SLAVE` 410
4.11.9 FAQ da Replicação 410
4.11.10 Problemas com Replicação 416
4.11.11 Relatando Problemas de Replicação 417

5 Otimização do MySQL 418

5.1 Visão Geral da Otimização 418
5.1.1 Limitações do Projeto MySQL/Trocas 418
5.1.2 Portabilidade 419
5.1.3 Para que Utilizamos o MySQL? 420
5.1.4 O Pacote de Benchmark do MySQL 421
5.1.5 Utilizando seus Próprios Benchmarks 422
5.2 Otimizando `SELECT`s e Outras Consultas 423
5.2.1 Sintaxe de `EXPLAIN` (Obter informações sobre uma `SELECT`) 424
5.2.2 Estimando o Desempenho de uma Consulta 431
5.2.3 Velocidade das Consultas que Utilizam `SELECT` . . 431
5.2.4 Como o MySQL Otimiza Cláusulas `WHERE` 432
5.2.5 Como o MySQL Otimiza `IS NULL` 433
5.2.6 Como o MySQL Otimiza Cláusulas `DISTINCT` . . . 434
5.2.7 Como o MySQL Otimiza `LEFT JOIN` e `RIGHT JOIN` 435
5.2.8 Como o MySQL Otimiza Cláusulas `ORDER BY` . . . 436
5.2.9 Como o MySQL Otimiza Cláusulas `LIMIT` 437
5.2.10 Performance das Consultas que Utilizam `INSERT` 438
5.2.11 Performance das Consultas que Utilizam `UPDATE` 440
5.2.12 Performance das Consultas que Utilizam `DELETE` 440
5.2.13 Mais Dicas sobre Otimizações 440
5.3 Detalhes sobre Locks 443
5.3.1 Como o MySQL Trava as Tabelas 443
5.3.2 Detalhes sobre Lock de Tabelas 444
5.4 Otimizando a Estrutura de Banco de Dados 446
5.4.1 Opções do Projeto 446
5.4.2 Deixando os Dados com o Menor Tamanho Possível 446
5.4.3 Como o MySQL Utiliza Índices 447
5.4.4 Índices de Colunas 449
5.4.5 Índices de Múltiplas Colunas 450
5.4.6 Como o MySQL Conta as Tabelas Abertas 451
5.4.7 Como o MySQL Abre e Fecha as Tabelas 451
5.4.8 Desvantagem em Criar um Número Grande de Tabelas no Mesmo Banco de Dados 452
5.5 Otimizando o Servidor MySQL 453
5.5.1 Sintonia dos Parâmetros em Tempo de Sistema/Compilação e na Inicialização 453
5.5.2 Parâmetros de Sintonia do Servidor 453
5.5.3 Como a Compilação e a Ligação Afetam a Velocidade do MySQL 456
5.5.4 Como o MySQL Utiliza a Memória 457
5.5.5 Como o MySQL Utiliza o DNS 459

5.5.6 Sintaxe de SET 459
5.6 Detalhes de Disco 464
5.6.1 Utilizando Links Simbólicos 465
5.6.1.1 Utilizando Links Simbólicos para Bancos de Dados 466
5.6.1.2 Utilizando Links Simbólicos para Tabelas 466
5.6.1.3 Usando Links Simbólicos para Bancos de Dados no Windows 467

6 Referência de Linguagem do MySQL 469

6.1 Estrutura da Linguagem 469
6.1.1 Literais: Como Gravar Strings e Numerais 469
6.1.1.1 Strings 469
6.1.1.2 Números 471
6.1.1.3 Valores Hexadecimais 471
6.1.1.4 Valores NULL 471
6.1.2 Nomes de Banco de dados, Tabela, Índice, Coluna e Alias 472
6.1.3 Caso Sensitivo nos Nomes 473
6.1.4 Variáveis de Usuário 474
6.1.5 Variáveis de Sistema 475
6.1.6 Sintaxe de Comentários 478
6.1.7 Tratamento de Palavras Reservadas no MySQL 479
6.2 Tipos de Campos 481
6.2.1 Tipos Numéricos 487
6.2.2 Tipos de Data e Hora 489
6.2.2.1 Assuntos referentes ao ano 2000 (Y2K) e Tipos de Data 490
6.2.2.2 Os Tipos DATETIME, DATE e TIMESTAMP 491
6.2.2.3 O Tipo TIME 495
6.2.2.4 O Tipo YEAR 496
6.2.3 Tipos String 496
6.2.3.1 Os Tipos CHAR e VARCHAR 497
6.2.3.2 Os Tipos BLOB e TEXT 497
6.2.3.3 O Tipo ENUM 499
6.2.3.4 O Tipo SET 500
6.2.4 Escolhendo o Tipo Correto para uma Coluna ... 501
6.2.5 Usando Tipos de Colunas de Outros Mecanismos de Banco de Dados 501
6.2.6 Exigências de Armazenamento dos Tipos de Coluna 502
6.3 Funções para Uso em Cláusulas SELECT e WHERE 503
6.3.1 Operadores e Funções de Tipos não Especificados 504
6.3.1.1 Parenteses 504

6.3.1.2 Operadores de Comparação 504
6.3.1.3 Operadores Logicos 508
6.3.1.4 Funções de Fluxo de Controle 510
6.3.2 Funções String 511
6.3.2.1 Funções de Comparação de Strings 519
6.3.2.2 Caso Sensitivo 521
6.3.3 Funções Numéricas 522
6.3.3.1 Operações Aritiméticas 522
6.3.3.2 Funções Matematicas 523
6.3.4 Funções de Data e Hora 529
6.3.5 Funções de Conversão 543
6.3.6 Outras Funções 545
6.3.6.1 Funções Binárias 546
6.3.6.2 Funções Diversas 546
6.3.7 Funções e Modificadores para Usar com Cláusulas GROUP BY 555
6.3.7.1 Funções GROUP BY 555
6.3.7.2 Modificadores GROUP BY 558
6.3.7.3 GROUP BY com Campos Escondidos 561
6.4 Manipulação de Dados: SELECT, INSERT, UPDATE e DELETE 562
6.4.1 Sintaxe SELECT 562
6.4.1.1 Sintaxe JOIN 567
6.4.1.2 Sintaxe UNION 569
6.4.2 Sintaxe de Subquery 569
6.4.2.1 A Subquery como um Operando Escalar 570
6.4.2.2 Comparações Usando Subquery 571
6.4.2.3 Subqueries with ANY, IN, and SOME 572
6.4.2.4 Subqueries with ALL 572
6.4.2.5 Correlated Subqueries 573
6.4.2.6 EXISTS and NOT EXISTS 573
6.4.2.7 Row Subqueries 574
6.4.2.8 Subqueries in the FROM clause 574
6.4.2.9 Subquery Errors 575
6.4.2.10 Optimizing Subqueries 576
6.4.2.11 Rewriting Subqueries for Earlier MySQL Versions 578
6.4.3 Sintaxe INSERT 579
6.4.3.1 Sintaxe INSERT ... SELECT 581
6.4.3.2 Sintaxe INSERT DELAYED 582
6.4.4 Sintaxe UPDATE 584
6.4.5 Sintaxe DELETE 585
6.4.6 Sintaxe TRUNCATE 587
6.4.7 Sintaxe REPLACE 587
6.4.8 Sintaxe LOAD DATA INFILE 588
6.4.9 Sintaxe HANDLER 595
6.4.10 Sintaxe DO 597

6.5 Definição de Dados: `CREATE`, `DROP` e `ALTER` 597
6.5.1 Sintaxe `CREATE DATABASE` 597
6.5.2 Sintaxe `DROP DATABASE` 597
6.5.3 Sintaxe `CREATE TABLE` 598
6.5.3.1 Alteração de Especificações de Colunas 607
6.5.4 Sintaxe `ALTER TABLE` 608
6.5.5 Sintaxe `RENAME TABLE` 612
6.5.6 Sintaxe `DROP TABLE` 612
6.5.7 Sintaxe `CREATE INDEX` 613
6.5.8 Sintaxe `DROP INDEX` 614
6.6 Comandos Utilitários Básicos do Usuário MySQL 614
6.6.1 Sintaxe `USE` 614
6.6.2 Sintaxe `DESCRIBE` (Obtem Informações Sobre Colunas) 614
6.7 Comandos Transacionais e de Lock do MySQL 615
6.7.1 Sintaxe de `START TRANSACTION`, `COMMIT` e `ROLLBACK` 615
6.7.2 Instruções que Não Podem Ser Desfeitas 615
6.7.3 Instruções que Fazem um Commit Implicito 616
6.7.4 Sintaxe de `SAVEPOINT` e `ROLLBACK TO SAVEPOINT` 616
6.7.5 Sintaxe `LOCK TABLES` e `UNLOCK TABLES` 617
6.7.6 Sintaxe `SET TRANSACTION` 619
6.8 Pesquisa Full-text no MySQL 619
6.8.1 Restrições Full-text 623
6.8.2 Ajuste Fino de Pesquisas Full-text no MySQL .. 624
6.8.3 TODO de Pesquisas Full-text 625
6.9 Cache de Consultas do MySQL 625
6.9.1 Como a Cache de Consultas Opera 626
6.9.2 Configuração da Cache de Consultas 627
6.9.3 Opções da Cache de Consultas na `SELECT` 628
6.9.4 Estado e Manutenção da Cache de Consultas ... 628

7 Tipos de Tabela do MySQL 630

7.1 Tabelas `MyISAM` 630
7.1.1 Espaço Necessário para Chaves 633
7.1.2 Formatos de Tabelas `MyISAM` 634
7.1.2.1 Características de Tabelas Estáticas (Tamanho Fixo) 634
7.1.2.2 Características de Tabelas Dinâmicas .. 634
7.1.2.3 Características de Tabelas Compactadas 635
7.1.3 Problemas com Tabelas `MyISAM` 636
7.1.3.1 Tabelas `MyISAM` Corrompidas 636
7.1.3.2 O Cliente está usando a tabela ou não a fechou de forma apropriada 637
7.2 Tabelas `MERGE` 637

7.2.1 Problemas com Tabelas MERGE 640
7.3 Tabelas ISAM 641
7.4 Tabelas HEAP 641
7.5 Tabelas InnoDB 643
7.5.1 Visão Geral de Tabelas InnoDB 643
7.5.2 InnoDB no MySQL Versão 3.23 643
7.5.3 Opções de Inicialização do InnoDB 644
7.5.4 Criando Tablespaces no InnoDB 651
7.5.4.1 Se Alguma Coisa Der Errado Na Criação Do Banco de Dados 652
7.5.5 Criando Tabelas InnoDB 653
7.5.5.1 Convertendo Tabelas MyISAM para InnoDB 653
7.5.5.2 Restrições FOREIGN KEY 654
7.5.5.3 Multiplos tablespaces - colocando cada tabela em seu próprio arquivo .ibd 656
7.5.6 Adicionando e Removendo Arquivos de Dados e Log do InnoDB 658
7.5.7 Fazendo Backup e Recuperando um Banco de Dados InnoDB 658
7.5.7.1 Forçando a recuperação 660
7.5.7.2 Ponto de Verificação 661
7.5.8 Movendo um Banco de Dados InnoDB para Outra Máquina 661
7.5.9 Modelo Transacional do InnoDB 662
7.5.9.1 InnoDB e SET ... TRANSACTION ISOLATION LEVEL 662
7.5.9.2 Leitura Consistente sem Lock 663
7.5.9.3 Lock de Leitura SELECT ... FOR UPDATE e SELECT ... LOCK IN SHARE MODE 664
7.5.9.4 Lock da Chave Seguinte: Evitando Problemas com Fantasmas 665
7.5.9.5 Locks Definidos por Diferentes Instruções SQL no InnoDB 665
7.5.9.6 Detecção de Deadlock e Rollback 666
7.5.9.7 Um Exemplo de Como a Leitura Consistente Funciona no InnoDB 667
7.5.9.8 Como lidar com deadlocks? 668
7.5.10 Dicas de Ajuste de Desempenho 669
7.5.10.1 SHOW INNODB STATUS e o Monitor InnoDB 670
7.5.11 Implementação de Multi-versioning 672
7.5.12 Estrutura de Tabelas e Índices 673
7.5.12.1 Estrutura Física do Índice 674
7.5.12.2 Buffer de Inserção 674
7.5.12.3 Índices Hash Adaptativos 675
7.5.12.4 Estrutura dos Registros Físicos 675

7.5.12.5 Como Funciona uma Coluna `AUTO_INCREMENT` no InnoDB 675
7.5.13 Gerenciamento do Espaço de Arquivos e E/S de Disco 676
7.5.13.1 E/S de Disco 676
7.5.13.2 Gerenciamento do Espaço de Arquivo 677
7.5.13.3 Desfragmentando uma Tabela 678
7.5.14 Tratando Erros 678
7.5.15 Restrições em Tabelas InnoDB 678
7.5.16 Histórico de Alterações do InnoDB 680
7.5.16.1 MySQL/InnoDB-4.1.1, December 4, 2003 680
7.5.16.2 MySQL/InnoDB-4.0.16, October 22, 2003 680
7.5.16.3 MySQL/InnoDB-3.23.58, September 15, 2003 680
7.5.16.4 MySQL/InnoDB-4.0.15, September 10, 2003 680
7.5.16.5 MySQL/InnoDB-4.0.14, Junho de 2003 681
7.5.16.6 MySQL/InnoDB-3.23.57, June 20, 2003 682
7.5.16.7 MySQL/InnoDB-4.0.13, 20 de Maio de 2003 682
7.5.16.8 MySQL/InnoDB-4.1.0, 03 de Abril de 2003 683
7.5.16.9 MySQL/InnoDB-3.23.56, 17 de Março de 2003 683
7.5.16.10 MySQL/InnoDB-4.0.12, 18 Março de 2003 684
7.5.16.11 MySQL/InnoDB-4.0.11, 25 de Fevereiro de 2003 684
7.5.16.12 MySQL/InnoDB-4.0.10, 04 de Fevereiro de 2003 684
7.5.16.13 MySQL/InnoDB-3.23.55, 24 de Janeiro de 2003 685
7.5.16.14 MySQL/InnoDB-4.0.9, 14 de Janeiro de 2003 685
7.5.16.15 MySQL/InnoDB-4.0.8, 07 de Janeiro de 2003 686
7.5.16.16 MySQL/InnoDB-4.0.7, 26 de Dezembro de 2002 686
7.5.16.17 MySQL/InnoDB-4.0.6, 19 de Dezembro de 2002 686
7.5.16.18 MySQL/InnoDB-3.23.54, 12 de Dezembro de 2002 687

7.5.16.19 MySQL/InnoDB-4.0.5, 18 de Novembro de 2002 687
7.5.16.20 MySQL/InnoDB-3.23.53, 09 de Outubro de 2002 688
7.5.16.21 MySQL/InnoDB-4.0.4, 02 de Outubro de 2002 689
7.5.16.22 MySQL/InnoDB-4.0.3, 28 de Agosto de 2002 690
7.5.16.23 MySQL/InnoDB-3.23.52, 16 de Agosto de 2002 690
7.5.16.24 MySQL/InnoDB-4.0.2, 10 de Julho de 2002 692
7.5.16.25 MySQL/InnoDB-3.23.51, 12 de Junho de 2002 692
7.5.16.26 MySQL/InnoDB-3.23.50, 23 de Abril de 2002 692
7.5.16.27 MySQL/InnoDB-3.23.49, 17 de Fevereiro de 2002 693
7.5.16.28 MySQL/InnoDB-3.23.48, 09 de Fevereiro de 2002 693
7.5.16.29 MySQL/InnoDB-3.23.47, 28 de Dezembro de 2001 694
7.5.16.30 MySQL/InnoDB-4.0.1, 23 de Dezembro de 2001 695
7.5.16.31 MySQL/InnoDB-3.23.46, 30 de Novembro de 2001 695
7.5.16.32 MySQL/InnoDB-3.23.45, 23 de Novembro de 2001 695
7.5.16.33 MySQL/InnoDB-3.23.44, 02 de Novembro de 2001 696
7.5.16.34 MySQL/InnoDB-3.23.43, 04 de Outubro de 2001 696
7.5.16.35 MySQL/InnoDB-3.23.42, 09 de Setembro de 2001 697
7.5.16.36 MySQL/InnoDB-3.23.41, 13 de Agosto de 2001 697
7.5.16.37 MySQL/InnoDB-3.23.40, 16 de Julho de 2001 697
7.5.16.38 MySQL/InnoDB-3.23.39, 13 de Junho de 2001 697
7.5.16.39 MySQL/InnoDB-3.23.38, 12 de Maio de 2001 697
7.5.17 Informações de Contato do `InnoDB` 697
7.6 Tabelas `BDB` ou `BerkeleyDB` 698
7.6.1 Visão Geral de Tabelas `BDB` 698
7.6.2 Instalando `BDB` 698
7.6.3 Opções de Inicialização do `BDB` 699
7.6.4 Caracterísitcas de Tabelas `BDB`: 700

7.6.5 Itens a serem corrigidos no BDB num futuro próximo: 701
7.6.6 Sistemas operacionais suportados pelo BDB 701
7.6.7 Restrições em Tabelas BDB 702
7.6.8 Erros Que Podem Ocorrer Usando Tabelas BDB .. 702

8 Introdução ao MaxDB 704

8.1 Historia do MaxDB 704
8.2 Licenciamento e Suporte 704
8.3 Conceitos Básicos do MaxDB 704
8.4 Diferenças de Recursos entre o MaxDB e o MySQL 704
8.5 Interoperability Features between MaxDB and MySQL ... 705
8.6 Links Relacionados ao MaxDB 705
8.7 Palavras Reservadas no MaxDB 706
8.8 Funções 708
8.9 Tipos de Colunas 709

9 Conjunto de Caracteres Nacionais e Unicode 710

9.1 Conjuntos de Caracteres e Collations em Geral 710
9.2 Conjunto de Caracteres e Collations no MySQL 711
9.3 Determinando o Conjunto de Caracteres e Collation Padrões 711
9.3.1 Conjunto de Caracteres e Collations do Servidor 711
9.3.2 Conjunto de Caracteres e Collation de Banco de Dados 712
9.3.3 O Conjunto de Caracteres e Collations de Tabela 713
9.3.4 Conjunto de Caracteres e Collation de Colunas .. 713
9.3.5 Exemplos de Atribuições de Conjuntos de Caracteres e Collation 714
9.3.6 Conjunto de Caracteres e Collation de Conexão 715
9.3.7 Conjunto de Caracteres e Collation de Caracter de String Literal 716
9.3.8 Cláusula COLLATE em Várias Partes de uma Consulta SQL 717
9.3.9 Precedência da Cláusula COLLATE 717
9.3.10 Operador BINARY 718
9.3.11 Alguns Casos Especiais Onde a Determinação da Collation e Trabalhosa 718
9.3.12 Collations Devem Ser para o Conjunto de Caracteres Certo 719
9.3.13 Um exemplo do Efeito da Collation 719
9.4 Operações Afetadas pelo Suporte a Conjunto de Caracteres 719

9.4.1 Strings de Resultados 720
9.4.2 CONVERT() 720
9.4.3 CAST() 720
9.4.4 SHOW CHARACTER SET 721
9.4.5 SHOW COLLATION 721
9.4.6 SHOW CREATE DATABASE 722
9.4.7 SHOW FULL COLUMNS 722
9.5 Suporte Unicode 722
9.6 UTF8 para Metdados 723
9.7 Compatibilidade com Outros SGBDs 724
9.8 Novo Formato do Arquivo de Configuração do Conjunto de Caracteres 724
9.9 Conjunto de Caracteres Nacional 724
9.10 Atualizando para o MySQL 4.0 725
9.10.1 Conjunto de Caracteres do MySQL e o Par/Conjunto de Caracter/Collation Correspondente do MySQL 4.1 726
9.11 Os conjuntos de Caracteres e Collations que o MySQL Suporta 726
9.11.1 O Conjunto de Caracteres Unicode 728
9.11.2 Conjunto de Caracteres para Plataformas Específicas 728
9.11.3 Conjunto de Caracteres do Sul da Europa e Oriente Médio 728
9.11.4 Os Conjuntos de Caracteres Asiáticos 728
9.11.5 Os Conjuntos de Caracteres Bálticos 729
9.11.6 Os Conjuntos de Caracteres Cirílicos 729
9.11.7 O Conjunto de Caracteres da Europa Central . . 730
9.11.8 Os Conjuntos de Caracteres da Europa Ocidental 731

10 Extensões Espacias em MySQL 733

10.1 Introdução 733
10.2 O Modelo Geomátrico OpenGIS 733
10.2.1 A Hierarquia da Classe Geometry 734
10.2.2 Classe Geometry 735
10.2.3 Classe Point 736
10.2.4 Classe Curve 736
10.2.5 Classe LineString 737
10.2.6 Classe Surface 737
10.2.7 Classe Polygon 737
10.2.8 Classe GeometryCollection 738
10.2.9 Classe MultiPoint 738
10.2.10 Classe MultiCurve 738
10.2.11 Classe MultiLineString (Multi Linhas) 739
10.2.12 Classe MultiSurface (Multi Superfícies) 739
10.2.13 Classe MultiPolygon (Multi Polígonos) 739
10.3 Formatos de Dados Espaciais Suportados 740

10.3.1 Formato Well-Known Text (WKT) 740
10.3.2 Formato Well-Known Binary (WKB) 741
10.4 Criando um Banco de Dados MySQL Habilitado Espacialmente 741
10.4.1 Tipos de Dados Espaciais do MySQL 742
10.4.2 Criando Valores Espaciais 742
10.4.2.1 Criando Valores Geometry Usando Funções WKT 742
10.4.2.2 Criando Valores Geometry Usando Funções WKB 743
10.4.2.3 Criando uma Valor de Geometira Usando Funções Específicas do MySQL 744
10.4.3 Criando Colunas Espaciais 745
10.4.4 Entrando com Dados em Colunas Espaciais 746
10.4.5 Buscando Dados Espaciais 747
10.4.5.1 Buscando Dados Espaciais em um Formato Interno 747
10.4.5.2 Buscando Dados Espaciais no Formato WKT 747
10.4.5.3 Buscando Dados Espaciais no Formato WKB 747
10.5 Analisando Informação Espacial 747
10.5.1 Funções Para Converter Geometrias Entre Formatos Diferentes 748
10.5.2 Funções de Análise das Propriedades de `Geometry` 748
10.5.2.1 Funções de Análise das Propriedades de `Geometry` em Geral 749
10.5.2.2 Funções de Análise das Propriedades de `Point` 750
10.5.2.3 Funções de Análise das Propriedades de `LineString` 751
10.5.2.4 Funções de Análise das Propriedades de `MultiLineString` 752
10.5.2.5 Funções de Análise das Propriedades de `Polygon` 753
10.5.2.6 Funções de Análise das Propriedades de `MultiPolygon` 754
10.5.2.7 Funções de Análise das Propriedades de `GeometryCollection` 754
10.5.3 Funções Que Criam Novas Geometrias de Outras Existentes 755
10.5.3.1 Funções de Geometria Que Produzem Novas Geometrias 755
10.5.3.2 Operadores Espaciais 755
10.5.4 Funções Para Testar Relações Espaciais Entre Objetos Geométricos 756

10.5.5 Relações de Retângulo de Limite Mínimo (Minimal Bounding Rectangles - MBR) em Geometrias 756
10.5.6 Funções que Testam Relacionamentos Espaciais Entre Geometrias 757
10.6 Otimizando Análises Espaciais 758
10.6.1 Criando Índices Espaciais 758
10.6.2 Usando Índice Espacial 759
10.7 Compatibilidade e Conformidade com o MySQL 761
10.7.1 Recursos GIS Que Ainda Não Estão Implementados 761

11 Stored Procedures e Funções 763

11.1 Sintaxe de Stored Procedure 763
11.1.1 Manutenção de Stored Procedures 764
11.1.1.1 `CREATE PROCEDURE` e `CREATE FUNCTION` 764
11.1.1.2 `ALTER PROCEDURE` e `ALTER FUNCTION` .. 766
11.1.1.3 `DROP PROCEDURE` e `DROP FUNCTION` 766
11.1.1.4 `SHOW CREATE PROCEDURE` e `SHOW CREATE FUNCTION` 767
11.1.2 `SHOW PROCEDURE STATUS` e `SHOW FUNCTION STATUS` 767
11.1.3 `CALL` 767
11.1.4 `BEGIN ... END` Compound Statement 767
11.1.5 Instrução `DECLARE` 767
11.1.6 Variables in Stored Procedures 768
11.1.6.1 Variável Local `DECLARE` 768
11.1.6.2 Instrução Variável `SET` 768
11.1.6.3 Instrução `SELECT ... INTO` 768
11.1.7 Condições e Handlers 768
11.1.7.1 `DECLARE` Conditions 768
11.1.7.2 `DECLARE` Handlers 769
11.1.8 Cursors 770
11.1.8.1 Declaring Cursors 771
11.1.8.2 Cursor `OPEN` Statement 771
11.1.8.3 Cursor `FETCH` Statement 771
11.1.8.4 Cursor `CLOSE` Statement 771
11.1.9 Flow Control Constructs 771
11.1.9.1 `IF` Statement 771
11.1.9.2 `CASE` Statement 772
11.1.9.3 `LOOP` Statement 772
11.1.9.4 `LEAVE` Statement 772
11.1.9.5 `ITERATE` Statement 772
11.1.9.6 `REPEAT` Statement 773
11.1.9.7 `WHILE` Statement 773

12 Ferramentas de Clientes e APIs do MySQL 775

12.1 API C do MySQL 775
12.1.1 Tipos de Dados da API C 775
12.1.2 Visão Geral das Função da API C 778
12.1.3 Descrição das Funções da API C 783
12.1.3.1 `mysql_affected_rows()` 783
12.1.3.2 `mysql_change_user()` 784
12.1.3.3 `mysql_character_set_name()` 785
12.1.3.4 `mysql_close()` 786
12.1.3.5 `mysql_connect()` 786
12.1.3.6 `mysql_create_db()` 787
12.1.3.7 `mysql_data_seek()` 787
12.1.3.8 `mysql_debug()` 788
12.1.3.9 `mysql_drop_db()` 788
12.1.3.10 `mysql_dump_debug_info()` 789
12.1.3.11 `mysql_eof()` 790
12.1.3.12 `mysql_errno()` 791
12.1.3.13 `mysql_error()` 792
12.1.3.14 `mysql_escape_string()` 792
12.1.3.15 `mysql_fetch_field()` 793
12.1.3.16 `mysql_fetch_fields()` 793
12.1.3.17 `mysql_fetch_field_direct()` 794
12.1.3.18 `mysql_fetch_lengths()` 795
12.1.3.19 `mysql_fetch_row()` 796
12.1.3.20 `mysql_field_count()` 797
12.1.3.21 `mysql_field_seek()` 798
12.1.3.22 `mysql_field_tell()` 799
12.1.3.23 `mysql_free_result()` 799
12.1.3.24 `mysql_get_client_info()` 799
12.1.3.25 `mysql_get_client_version()` 800
12.1.3.26 `mysql_get_host_info()` 800
12.1.3.27 `mysql_get_proto_info()` 801
12.1.3.28 `mysql_get_server_info()` 801
12.1.3.29 `mysql_get_server_version()` 801
12.1.3.30 `mysql_info()` 802
12.1.3.31 `mysql_init()` 803
12.1.3.32 `mysql_insert_id()` 803
12.1.3.33 `mysql_kill()` 804
12.1.3.34 `mysql_list_dbs()` 804
12.1.3.35 `mysql_list_fields()` 805
12.1.3.36 `mysql_list_processes()` 806
12.1.3.37 `mysql_list_tables()` 806
12.1.3.38 `mysql_num_fields()` 807
12.1.3.39 `mysql_num_rows()` 808
12.1.3.40 `mysql_options()` 809
12.1.3.41 `mysql_ping()` 811
12.1.3.42 `mysql_query()` 812

12.1.3.43 mysql_real_connect() 812
12.1.3.44 mysql_real_escape_string() 815
12.1.3.45 mysql_real_query() 816
12.1.3.46 mysql_reload() 817
12.1.3.47 mysql_row_seek() 818
12.1.3.48 mysql_row_tell() 818
12.1.3.49 mysql_select_db() 819
12.1.3.50 mysql_set_server_option() 819
12.1.3.51 mysql_shutdown() 820
12.1.3.52 mysql_sqlstate() 820
12.1.3.53 mysql_ssl_set() 821
12.1.3.54 mysql_stat() 821
12.1.3.55 mysql_store_result() 822
12.1.3.56 mysql_thread_id() 823
12.1.3.57 mysql_use_result() 824
12.1.3.58 mysql_warning_count() 825
12.1.3.59 mysql_commit() 825
12.1.3.60 mysql_rollback() 826
12.1.3.61 mysql_autocommit() 826
12.1.3.62 mysql_more_results() 826
12.1.3.63 mysql_next_result() 827
12.1.4 Instruções Preparadas da API C 828
12.1.5 Tipos de Dados de Instruções Preparadas da API C 828
12.1.6 Visão Geral das Funções de Instruções Preparadas da API C 831
12.1.7 Descrição das Funções de Instrução Preparada da API C 833
12.1.7.1 mysql_prepare() 833
12.1.7.2 mysql_param_count() 835
12.1.7.3 mysql_get_metadata() 835
12.1.7.4 mysql_bind_param() 836
12.1.7.5 mysql_execute() 837
12.1.7.6 mysql_stmt_affected_rows() 841
12.1.7.7 mysql_bind_result() 842
12.1.7.8 mysql_stmt_store_result() 843
12.1.7.9 mysql_stmt_data_seek() 844
12.1.7.10 mysql_stmt_row_seek() 844
12.1.7.11 mysql_stmt_row_tell() 845
12.1.7.12 mysql_stmt_num_rows() 845
12.1.7.13 mysql_fetch() 846
12.1.7.14 mysql_send_long_data() 851
12.1.7.15 mysql_stmt_close() 853
12.1.7.16 mysql_stmt_errno() 854
12.1.7.17 mysql_stmt_error() 854
12.1.7.18 mysql_stmt_sqlstate() 855
12.1.8 Tratando a Execução de Múltiplas Consultas na API C 855

12.1.9 Manipulando Valores de Data e Hora na API C 856
12.1.10 Descrição das Funções de Threads da API C .. 858
12.1.10.1 `my_init()` 858
12.1.10.2 `mysql_thread_init()` 858
12.1.10.3 `mysql_thread_end()` 858
12.1.10.4 `mysql_thread_safe()` 859
12.1.11 Descrição das Funções do Servidor Embutido da API C 859
12.1.11.1 `mysql_server_init()` 859
12.1.11.2 `mysql_server_end()` 860
12.1.12 Dúvidas e problemas comuns ao utilzar a API C 861
12.1.12.1 Porque Algumas Vezes `mysql_store_result()` Retorna `NULL` Após `mysql_query()` Returnar com Sucesso? 861
12.1.12.2 Que Resultados Posso Onbetr de uma Consulta? 861
12.1.12.3 Como Posso Obter a ID Única para a Última Linha Inserida? 861
12.1.12.4 Problemas com Ligação na API C ... 862
12.1.13 Construindo Programas Clientes 862
12.1.14 Como Fazer um Cliente em Threads 863
12.1.15 libmysqld, a Biblioteca do Servidor Embutido MySQL 864
12.1.15.1 Visão Geral da Biblioteca do Servidor MySQL Embutido 864
12.1.15.2 Compilando Programas com `libmysqld` 865
12.1.15.3 Restrições no Uso de um Servidor MySQL Embutido 865
12.1.15.4 Usando Arquivo de Opções com o Servidor Embutido 865
12.1.15.5 Itens a Fazer no Servidor Embutido (TODO) 866
12.1.15.6 Um Exemplo Simples de Servidor Embutido 866
12.1.15.7 Licensiando o Servidor Embutido ... 870
12.2 Suporte ODBC ao MySQL 870
12.2.1 Como Instalar o MyODBC 870
12.2.2 Como Preencher os Vários Campos no Programa de Administração do ODBC 871
12.2.3 Parâmetros de Conexão do MyODBC 872
12.2.4 Como Relatar Problemas com o MyODBC 873
12.2.5 Programas que Funcionam com MyODBC 874
12.2.6 Como Obter o Valor de uma Coluna `AUTO_INCREMENT` no ODBC 878
12.2.7 Relatando Problemas com MyODBC 879

12.3 Conectividade Java (JDBC) ao MySQL 880
12.4 API PHP do MySQL 880
12.4.1 Problemas Comuns com MySQL e PHP 880
12.5 API Perl do MySQL 880
12.5.1 DBI com DBD::mysql 881
12.5.2 A interface DBI 881
12.5.3 Mais Informações DBI/DBD 887
12.6 API C++ do MySQL 887
12.6.1 Borland C++ 887
12.7 API Python do MySQL 888
12.8 API Tcl do MySQL 888
12.9 Eiffel Wrapper do MySQL 888

13 Tratamento de Erros no MySQL 889
13.1 Erros Retornados 889

14 Estendendo o MySQL 896
14.1 MySQL Internals 896
14.1.1 Threads MySQL 896
14.1.2 Pacotes de Teste do MySQL 896
14.1.2.1 Executando o Pacote de Testes do MySQL 897
14.1.2.2 Extendendo o Pacote de Teste do MySQL 897
14.1.2.3 Relatando Bugs no Pacote de Teste do MySQL 898
14.2 Adicionando Novas Funções ao MySQL 899
14.2.1 Sintaxe CREATE FUNCTION/DROP FUNCTION 900
14.2.2 Adicionando Novas Funções Definidas Por Usuário 900
14.2.2.1 Sequência de Chamadas UDF para Funções Simples 902
14.2.2.2 Sequência de Chamadas UDF para Funções Agregadas 903
14.2.2.3 Processando Argumentos 904
14.2.2.4 Valor de Retorno e Tartamento de Erros 906
14.2.2.5 Compilando e Instalando Funções Definidas Por Usuário 906
14.2.3 Adicionando uma Nova Função Nativa 908
14.3 Adicionado Novos Procedimentos ao MySQL 909
14.3.1 Análise de Procedimento 909
14.3.2 Escrevendo um Procedimento 910

Apêndice A Problemas e Erros Comuns 911

A.1 Como Determinar o Que Está Causando Problemas 911
A.2 Erros Comuns Usando o MySQL 912
A.2.1 Erro: `Access Denied` 912
A.2.2 Erro: `MySQL server has gone away` 912
A.2.3 Erro: `Can't connect to [local] MySQL server` 913
A.2.4 Erro: `Client does not support authentication protocol` 915
A.2.5 Erro: `Host '...' is blocked` 916
A.2.6 Erro: `Too many connections` 916
A.2.7 Erro: `Some non-transactional changed tables couldn't be rolled back` 916
A.2.8 Erro: `Out of memory` 917
A.2.9 Erro: `Packet too large` 917
A.2.10 Erros de Comunicação / Comunicação Abortada 918
A.2.11 Erro: `The table is full` 919
A.2.12 Erro: `Can't create/write to file` 919
A.2.13 Erro no Cliente: `Commands out of sync` 920
A.2.14 Erro: `Ignoring user` 920
A.2.15 Erro: `Table 'xxx' doesn't exist` 920
A.2.16 Erro: `Can't initialize character set xxx` .. 921
A.2.17 Arquivo Não Encontrado 921
A.3 Assuntos Relacionados a Instalação 922
A.3.1 Problemas de Ligação com a Biblioteca do Cliente MySQL 922
A.3.2 Como Executar o MySQL Como Um Usuário Normal 923
A.3.3 Problemas com Permissões de Arquivos 924
A.4 Assuntos Relacionados a Administração 924
A.4.1 O Que Fazer Se o MySQL Continua Falhando .. 925
A.4.2 Como Recuperar uma Senha de Root Esquecida 927
A.4.3 Como o MySQL Trata de Discos Sem Espaço .. 928
A.4.4 Onde o MySQL Armazena Arquivos Temporários 928
A.4.5 Como Proteger ou AlterarHow to Protect or Change the MySQL Socket File '`/tmp/mysql.sock`' 929
A.4.6 Problemas Com Fuso Horário 930
A.5 Assuntos Relacionados a Consultas 930
A.5.1 Caso-Sensitivito em Pesquisas 930
A.5.2 Problemas Usando Colunas `DATE` 930
A.5.3 Problemas com Valores `NULL` 932
A.5.4 Problemas com `alias` 933
A.5.5 Deletando Linhas de Tabelas Relacionadas 933
A.5.6 Resolvendo Problemas Com Registros Não Encontrados 934

A.5.7 Problemas com Comparação de Ponto Flutuante 934
A.6 Assuntos Relacionados ao Otimizador 936
A.6.1 Camo evitar o varredura da tabela,,, ... 937
A.7 Assuntos Relacionados a Definições de Tabelas 937
A.7.1 Problemas com `ALTER TABLE` 937
A.7.2 Como Alterar a Ordem das Colunas em Uma Tabela 938
A.7.3 Problemas com TEMPORARY TABLE 938

Apêndice B Contribuição de Programas 940

B.1 APIs 940
B.2 Conversores 942
B.3 Utilitários 944

Apêndice C Colaboradores do MySQL 945

C.1 Desenvolvedores do MySQL 945
C.2 Coolaboradores do MySQL 948
C.3 Responsáveis pela Documentação e Tradução 952
C.4 Bibliotecas usadas e incluidas com o MySQL 953
C.5 Pacotes que suportam o MySQL 954
C.6 Ferramentas que são usadas para criar o MySQL 955
C.7 Responsáveis pelo Suporte do MySQL 955

Apêndice D Histórico de Alterações do MySQL 957

D.1 Alterações na distribuição 5.0.0 (Development) 957
D.2 Alterações na distribuição 4.1.x (Alpha) 957
D.2.1 Alterações na distribuição 4.1.2 (not released yet) 958
D.2.2 Alterações na distribuição 4.1.1 (01 de Dez de 2003) 959
D.2.3 Alterações na distribuição 4.1.0 (03 Apr 2003: Alpha) 963
D.3 Alterações na distribuição 4.0.x (Production) 965
D.3.1 Alterações na distribuição 4.0.17 (not released yet) 966
D.3.2 Alterações na distribuição 4.0.16 (17 Out 2003) 968
D.3.3 Alterações na distribuição 4.0.15 (03 Sep 2003) 970
D.3.4 Alterações na distribuição 4.0.14 (18 Jul 2003) .. 974
D.3.5 Alterações na distribuição 4.0.13 (16 May 2003) 978
D.3.6 Alterações na distribuição 4.0.12 (15 Mar 2003: Production) 981

D.3.7 Alterações na distribuição 4.0.11 (20 Feb 2003) 983
D.3.8 Alterações na distribuição 4.0.10 (29 Jan 2003) 984
D.3.9 Alterações na distribuição 4.0.9 (09 Jan 2003) . . 985
D.3.10 Alterações na distribuição 4.0.8 (07 Jan 2003) 986
D.3.11 Alterações na distribuição 4.0.7 (20 Dec 2002) 986
D.3.12 Alterações na distribuição 4.0.6 (14 Dec 2002: Gamma) 987
D.3.13 Alterações na distribuição 4.0.5 (13 Nov 2002) 988
D.3.14 Alterações na distribuição 4.0.4 (29 Sep 2002) 990
D.3.15 Alterações na distribuição 4.0.3 (26 Aug 2002: Beta) 992
D.3.16 Alterações na distribuição 4.0.2 (01 Jul 2002) . . 994
D.3.17 Alterações na distribuição 4.0.1 (23 Dec 2001) 997
D.3.18 Alterações na distribuição 4.0.0 (Oct 2001: Alpha) 998
D.4 Alterações na distribuição 3.23.x (Recent; still supported) 1000
D.4.1 Alterações na distribuição 3.23.59 (not released yet) 1000
D.4.2 Alterações na distribuição 3.23.58 (11 Sep 2003) 1001
D.4.3 Alterações na distribuição 3.23.57 (06 Jun 2003) 1001
D.4.4 Alterações na distribuição 3.23.56 (13 Mar 2003) 1002
D.4.5 Alterações na distribuição 3.23.55 (23 Jan 2003) 1003
D.4.6 Alterações na distribuição 3.23.54 (05 Dec 2002) 1004
D.4.7 Alterações na distribuição 3.23.53 (09 Oct 2002) 1005
D.4.8 Alterações na distribuição 3.23.52 (14 Aug 2002) 1006
D.4.9 Alterações na distribuição 3.23.51 (31 May 2002) 1006
D.4.10 Alterações na distribuição 3.23.50 (21 Apr 2002) 1007
D.4.11 Alterações na distribuição 3.23.49 1008
D.4.12 Alterações na distribuição 3.23.48 (07 Feb 2002) 1008

D.4.13 Alterações na distribuição 3.23.47 (27 Dec 2001) 1009
D.4.14 Alterações na distribuição 3.23.46 (29 Nov 2001) 1009
D.4.15 Alterações na distribuição 3.23.45 (22 Nov 2001) 1010
D.4.16 Alterações na distribuição 3.23.44 (31 Oct 2001) 1010
D.4.17 Alterações na distribuição 3.23.43 (04 Oct 2001) 1011
D.4.18 Alterações na distribuição 3.23.42 (08 Sep 2001) 1012
D.4.19 Alterações na distribuição 3.23.41 (11 Aug 2001) 1013
D.4.20 Alterações na distribuição 3.23.40 1013
D.4.21 Alterações na distribuição 3.23.39 (12 Jun 2001) 1014
D.4.22 Alterações na distribuição 3.23.38 (09 May 2001) 1014
D.4.23 Alterações na distribuição 3.23.37 (17 Apr 2001) 1015
D.4.24 Alterações na distribuição 3.23.36 (27 Mar 2001) 1016
D.4.25 Alterações na distribuição 3.23.35 (15 Mar 2001) 1016
D.4.26 Alterações na distribuição 3.23.34a 1017
D.4.27 Alterações na distribuição 3.23.34 (10 Mar 2001) 1017
D.4.28 Alterações na distribuição 3.23.33 (09 Feb 2001) 1018
D.4.29 Alterações na distribuição 3.23.32 (22 Jan 2001: Production) 1019
D.4.30 Alterações na distribuição 3.23.31 (17 Jan 2001) 1019
D.4.31 Alterações na distribuição 3.23.30 (04 Jan 2001) 1020
D.4.32 Alterações na distribuição 3.23.29 (16 Dec 2000) 1021
D.4.33 Alterações na distribuição 3.23.28 (22 Nov 2000: Gamma) 1022
D.4.34 Alterações na distribuição 3.23.27 (24 Oct 2000) 1024
D.4.35 Alterações na distribuição 3.23.26 (18 Oct 2000) 1024
D.4.36 Alterações na distribuição 3.23.25 (29 Sep 2000) 1025
D.4.37 Alterações na distribuição 3.23.24 (08 Sep 2000) 1026

D.4.38 Alterações na distribuição 3.23.23 (01 Sep 2000) 1027
D.4.39 Alterações na distribuição 3.23.22 (31 Jul 2000) 1028
D.4.40 Alterações na distribuição 3.23.21 1028
D.4.41 Alterações na distribuição 3.23.20 1029
D.4.42 Alterações na distribuição 3.23.19 1029
D.4.43 Alterações na distribuição 3.23.18 1030
D.4.44 Alterações na distribuição 3.23.17 1030
D.4.45 Alterações na distribuição 3.23.16 1031
D.4.46 Alterações na distribuição 3.23.15 (May 2000: Beta) 1031
D.4.47 Alterações na distribuição 3.23.14 1032
D.4.48 Alterações na distribuição 3.23.13 1033
D.4.49 Alterações na distribuição 3.23.12 (07 Mar 2000) 1033
D.4.50 Alterações na distribuição 3.23.11 1034
D.4.51 Alterações na distribuição 3.23.10 1034
D.4.52 Alterações na distribuição 3.23.9 1034
D.4.53 Alterações na distribuição 3.23.8 (02 Jan 2000) 1035
D.4.54 Alterações na distribuição 3.23.7 (10 Dec 1999) 1036
D.4.55 Alterações na distribuição 3.23.6 1036
D.4.56 Alterações na distribuição 3.23.5 (20 Oct 1999) 1037
D.4.57 Alterações na distribuição 3.23.4 (28 Sep 1999) 1038
D.4.58 Alterações na distribuição 3.23.3 1038
D.4.59 Alterações na distribuição 3.23.2 (09 Aug 1999) 1039
D.4.60 Alterações na distribuição 3.23.1 1040
D.4.61 Alterações na distribuição 3.23.0 (05 Aug 1999: Alpha) 1040
D.5 Alterações na distribuição 3.22.x (Old; discontinued) . . . 1042
D.5.1 Alterações na distribuição 3.22.35 1042
D.5.2 Alterações na distribuição 3.22.34 1042
D.5.3 Alterações na distribuição 3.22.33 1042
D.5.4 Alterações na distribuição 3.22.32 (14 Feb 2000) 1042
D.5.5 Alterações na distribuição 3.22.31 1043
D.5.6 Alterações na distribuição 3.22.30 1043
D.5.7 Alterações na distribuição 3.22.29 (02 Jan 2000) 1043
D.5.8 Alterações na distribuição 3.22.28 (20 Oct 1999) 1043
D.5.9 Alterações na distribuição 3.22.27 1043

D.5.10 Alterações na distribuição 3.22.26 (16 Sep 1999) 1044
D.5.11 Alterações na distribuição 3.22.25 1044
D.5.12 Alterações na distribuição 3.22.24 (05 Jul 1999) 1044
D.5.13 Alterações na distribuição 3.22.23 (08 Jun 1999) 1044
D.5.14 Alterações na distribuição 3.22.22 (30 Apr 1999) 1045
D.5.15 Alterações na distribuição 3.22.21 1045
D.5.16 Alterações na distribuição 3.22.20 (18 Mar 1999) 1045
D.5.17 Alterações na distribuição 3.22.19 (Mar 1999: Production) 1045
D.5.18 Alterações na distribuição 3.22.18 1045
D.5.19 Alterações na distribuição 3.22.17 1046
D.5.20 Alterações na distribuição 3.22.16 (Feb 1999: Gamma) 1046
D.5.21 Alterações na distribuição 3.22.15 1046
D.5.22 Alterações na distribuição 3.22.14 1047
D.5.23 Alterações na distribuição 3.22.13 1047
D.5.24 Alterações na distribuição 3.22.12 1047
D.5.25 Alterações na distribuição 3.22.11 1048
D.5.26 Alterações na distribuição 3.22.10 1048
D.5.27 Alterações na distribuição 3.22.9 1049
D.5.28 Alterações na distribuição 3.22.8 1049
D.5.29 Alterações na distribuição 3.22.7 (Sep 1998: Beta) 1050
D.5.30 Alterações na distribuição 3.22.6 1050
D.5.31 Alterações na distribuição 3.22.5 1051
D.5.32 Alterações na distribuição 3.22.4 1052
D.5.33 Alterações na distribuição 3.22.3 1053
D.5.34 Alterações na distribuição 3.22.2 1053
D.5.35 Alterações na distribuição 3.22.1 (Jun 1998: Alpha) 1054
D.5.36 Alterações na distribuição 3.22.0 1054
D.6 Alterações na distribuição 3.21.x 1056
D.6.1 Alterações na distribuição 3.21.33 1056
D.6.2 Alterações na distribuição 3.21.32 1056
D.6.3 Alterações na distribuição 3.21.31 1056
D.6.4 Alterações na distribuição 3.21.30 1057
D.6.5 Alterações na distribuição 3.21.29 1057
D.6.6 Alterações na distribuição 3.21.28 1057
D.6.7 Alterações na distribuição 3.21.27 1058
D.6.8 Alterações na distribuição 3.21.26 1058
D.6.9 Alterações na distribuição 3.21.25 1058
D.6.10 Alterações na distribuição 3.21.24 1059
D.6.11 Alterações na distribuição 3.21.23 1059

D.6.12 Alterações na distribuição 3.21.22 1059
D.6.13 Alterações na distribuição 3.21.21a 1060
D.6.14 Alterações na distribuição 3.21.21 1060
D.6.15 Alterações na distribuição 3.21.20 1060
D.6.16 Alterações na distribuição 3.21.19 1061
D.6.17 Alterações na distribuição 3.21.18 1061
D.6.18 Alterações na distribuição 3.21.17 1061
D.6.19 Alterações na distribuição 3.21.16 1062
D.6.20 Alterações na distribuição 3.21.15 1062
D.6.21 Alterações na distribuição 3.21.14b 1063
D.6.22 Alterações na distribuição 3.21.14a 1063
D.6.23 Alterações na distribuição 3.21.13 1063
D.6.24 Alterações na distribuição 3.21.12 1064
D.6.25 Alterações na distribuição 3.21.11 1064
D.6.26 Alterações na distribuição 3.21.10 1065
D.6.27 Alterações na distribuição 3.21.9 1065
D.6.28 Alterações na distribuição 3.21.8 1065
D.6.29 Alterações na distribuição 3.21.7 1066
D.6.30 Alterações na distribuição 3.21.6 1066
D.6.31 Alterações na distribuição 3.21.5 1066
D.6.32 Alterações na distribuição 3.21.4 1067
D.6.33 Alterações na distribuição 3.21.3 1067
D.6.34 Alterações na distribuição 3.21.2 1068
D.6.35 Alterações na distribuição 3.21.0 1068
D.7 Alterações na distribuição 3.20.x 1069
D.7.1 Alterações na distribuição 3.20.18 1069
D.7.2 Alterações na distribuição 3.20.17 1070
D.7.3 Alterações na distribuição 3.20.16 1071
D.7.4 Alterações na distribuição 3.20.15 1071
D.7.5 Alterações na distribuição 3.20.14 1071
D.7.6 Alterações na distribuição 3.20.13 1072
D.7.7 Alterações na distribuição 3.20.11 1072
D.7.8 Alterações na distribuição 3.20.10 1073
D.7.9 Alterações na distribuição 3.20.9 1073
D.7.10 Alterações na distribuição 3.20.8 1073
D.7.11 Alterações na distribuição 3.20.7 1073
D.7.12 Alterações na distribuição 3.20.6 1074
D.7.13 Alterações na distribuição 3.20.3 1075
D.7.14 Alterações na distribuição 3.20.0 1076
D.8 Alterações na distribuição 3.19.x 1076
D.8.1 Alterações na distribuição 3.19.5 1076
D.8.2 Alterações na distribuição 3.19.4 1077
D.8.3 Alterações na distribuição 3.19.3 1077

Apêndice E Portando para Outros Sistemas 1078

E.1 Depurando um Servidor MySQL 1079
E.1.1 Compilando o MYSQL para Depuração 1079
E.1.2 Criando Arquivos Trace (Rastreamento) 1080
E.1.3 Depurando o mysqld no gdb 1081
E.1.4 Usando Stack Trace 1082
E.1.5 Usando Arquivos de Log para Encontrar a Causa dos Erros no mysqld 1083
E.1.6 Fazendo um Caso de Teste Se Ocorre um Corrompimento de Tabela 1084
E.2 Depurando um cliente MySQL. 1085
E.3 O Pacote DBUG 1085
E.4 Métodos de Lock 1087
E.5 Comentários Sobre Threads RTS 1088
E.6 Diferença en Entre Alguns Pacotes de Threads 1090

Apêndice F Variáveis de Ambientes do MySQL 1092

Apêndice G Sintaxe de Expressões Regulares do MySQL 1093

Apêndice H GPL - Licença Pública Geral do GNU 1096

Índices dos Comandos, Tipos e Funções SQL 1102

Concept Index 1111

1 Informações Gerais

O programa MySQL ® é um servidor robusto de bancos de dados SQL (Structured Query Language - Linguagem Estruturada para Pesquisas) muito rápido, multi-tarefa e multi-usuário. O Servidor MySQL pode ser usado em sistemas de produção com alta carga e missão crítica bem como pode ser embutido em programa de uso em massa. MySQL é uma marca registrada da MySQL AB.

O programa MySQL é de Licença Dupla. Os usuários podem escolher entre usar o programa MySQL como um produto Open Source/Free Software sob os termos da GNU General Public License (http://www.fsf.org/licenses/) ou podem comprar uma licença comercial padrão da MySQL AB. Veja Seção 1.4 [Licensing and Support], Página 17.

O site web do MySQL (http://www.mysql.com/) dispõe das últimas informações sobre o programa MySQL.

A seguinte lista descreve algumas seções de particular interesse neste manual:

- Para informações sobre a empresa por trás do Servidor do Banco de Dados MySQL, veja Seção 1.3 [What is MySQL AB], Página 12.
- Para discussões das capacidades do Servidor do Banco de Dados MySQL, veja Seção 1.2.2 [Features], Página 6.
- Para instruções de instalação, veja Cap“ptexi tulo 2 [Installing], Página 60.
- Para dicas sobre a portabilidade do Servidor do Banco de Dados MySQL para novas arquiteturas ou sistemas operacionais, veja Apêndice E [Porting], Página 1078.
- Para informações sobre a atualização da versão 4.0, veja Seção 2.5.1 [Upgrading-from-4.0], Página 121.
- Para informações sobre a atualização da versão 3.23, veja Seção 2.5.2 [Upgrading-from-3.23], Página 123.
- Para informações sobre a atualização da versão 3.22, veja Seção 2.5.3 [Upgrading-from-3.22], Página 127.
- Para um tutorial de introdução ao Servidor do Banco de Dados MySQL, veja Cap“ptexi tulo 3 [Tutorial], Página 168.
- Para exemplos de SQL e informações sobre benchmarks, veja o diretório de benchmarks (‘sql-bench’ na distribuição).
- Para o histórico de novos recursos e correções de erros, veja Apêndice D [News], Página 957.
- Para uma lista de erros atualmente conhecidos e mal-funcionamento, veja Seção 1.8.6 [Bugs], Página 54.
- Para projetos futuros, veja Seção 1.6 [TODO], Página 27.
- Para ver a lista de todos os colaboradores deste projeto, veja Apêndice C [Credits], Página 945.

Importante:

Relatórios de erros (também chamados bugs), bem como dúvidas e comentários, devem ser enviados para a lista de email geral do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33. Veja Seção 1.7.1.3 [Bug reports], Página 36.

O script `mysqlbug` deve ser usado para gerar comunicados de erros no Unix. (A distribuição do Windows contém um arquivo '`mysqlbug.txt`' no diretório base que pode ser usado como um template para um relatório de erro.

Em distribuições fonte, o script `mysqlbug` pode ser encontrado no diretório '`scripts`'. Para distribuições binárias, o `mysqlbug` pode ser encontrado no diretório '`bin`' ('`/usr/bin`' para o pacote RMP do `servidor MySQL`.

Se você encontrou um erro de segurança no `Servidor MySQL`, você deve enviar um email para `security@mysql.com`.

1.1 Sobre Este Manual

Este é o manual de referência `MySQL`; ele documenta o `MySQL` até a versão 5.0.0-alpha. Mudanças funcionais são sempre indicadas com referência a versão, assim este manual também pode ser utiliado caso você esteja utilizando uma versão mais antiga do `MySQL` (como 3.23 ou 4.0-produção). Também a referências a versão 5.0 (desenvolvimento).

Sendo um manual de referência, ele não fornece instruções gerais sobre `SQL` ou conceitos de banco de dados relacionais.

Como o `Programa da Banco de Dados MySQL` está sob constante desenvolvimento, o manual também é atualizado freqüentemente. A versão mais recente deste manual está disponível em `http://www.mysql.com/documentation/` em diferentes formatos, incluindo HTML, PDF, e versões HLP do Windows.

O documento original é uma arquivo Texinfo. A versão HTML é produzida automaticamente usando uma versão modificada do `texi2html`. A versão texto e Info são produzidas com `makeinfo`. A versão PostScript é produzida usando `texi2dvi` e `dvips`. A versão PDF é produzida com `pdftex`.

Se você tiver dificuldades de encontrar informações no manual, você pode tentar nossa versão com busca em `http://www.mysql.com/doc/`.

Se você tiver qualquer sugestão a respeito de adições e correções neste manual, por favor envie-os para a equipe de documentação em `docs@mysql.com`.

Este manual foi inicialmente escrito por David Axmark e Michael (Monty) Widenius. Atualmente é mantido pela Equipe de Documentação da MySQL, que conta com Arjen Lentz, Paul DuBois e Stefan Hinz. Para outros colaboradores, veja Apêndice C [Credits], Página 945.

A traduçao deste manual foi feita por Daniel Coelho Teobaldo e Carlos Henrique Paulino sob a supervisão da EAC Software.

Os direitos autorais (2003) deste manual pertence a compania Sueca `MySQL AB`. Veja Seção 1.4.2 [Direitos Autorais], Página 17.

1.1.1 Convenções Usadas Neste Manual

Este manual utiliza algumas convenções tipográficas:

`constant` Fonte de largura fixa é usada para nomes de comandos e opções; instruções SQL; nomes de bancos de dados, tabelas e colunas; código C e Perl; e variáveis

de ambiente. Exemplo: "Para ver como o `mysqladmin` funciona, execute-o com a opção `--help`."

'`filename`'
: Fonte de largura fixa com aspas é usada para nomes e caminhos de arquivos. Exemplo: "A distribuição é instalada sobre o diretório '`/usr/local`'."

'`c`'
: Fonte de largura constante com aspas é também usada para indicar sequências de caracteres. Exemplo: "Para especificar um meta caracter, use o caractere '`%`'."

italic
: Fonte Itálica é usada para dar ênfase, *como aqui*.

boldface
: Fonte em negrito é usada em cabeçalhos de tabela e indicar **ênfase especial**.

Quando um comando deve ser executado por um programa, ele é indicado por um prompt antes do comando. Por exemplo, `shell>` indica um comando que é executado do seu shell atual e `mysql>` indica um comando que é executado no programa cliente `mysql`;

```
shell> digite um comando shell aqui
mysql> digite um comando mysql aqui
```

A "shell" é seu interpretador de comando. No Unix, ele é normalmente um programa como `sh` ou `csh`. No Windows, o equivalente é o `command.com` ou `cmd.exe`, normalmente executado como um console do Windows.

Comandos Shell são mostrados usando a sintaxe do Shell Bourne. Se você usa um shell do estilo `csh`, pode ser necessário alterar algum de seus comandos. Por exemplo, a sequência para configurar uma variável de ambiente e executar um comando se parece com o listado abaixo na sintaxe Bourne Shell:

```
shell> NOMEVAR=valor algum_comando
```

Para `csh` ou `tcsh`, execute a sequência desta forma:

```
shell> setenv NOMEVAR valor
shell> algum_comando
```

Frequentemente, nomes de bancos de dados, tabelas e colunas devem ser substituídos nos comandos. Para indicar que as substituições são necessárias, este manual usa `nome_db`, `nome_tbl` e `nome_col`. Por exemplo, você pode encontrar uma expressão assim:

```
mysql> SELECT nome_col FROM nome_bd.nome_tbl;
```

Isso significa que se você estiver trabalhando numa expressão similar, forneceria seu próprio nome de banco de dados, tabela e colunas, talvez assim:

```
mysql> SELECT nome_autor FROM biblio_bd.lista_autor;
```

SQL keywords não caso sensitivas e podem ser escritas em maiúscula ou minúscula. Este manual utiliza letras maiúsculas.

Em descrições de sintaxe, colchetes ('`[`' e '`]`') são usados para indicar palavras ou cláusulas opcionais. Por exemplo, na seguinte instrução, `IF EXISTS` é opcional:

```
DROP TABLE [IF EXISTS] nome_tbl
```

Quando elementos da sintaxe possuem mais de uma alternativa, elas são separados por barras verticais ('`|`'). Quando um menbro de um conjunto de opções **pode ser** escolhido, as alternativas são listadas em colchetes ('`[`' e '`]`'):

```
TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)
```

Quando um membro de um conjunto de opções **deve** ser selecionado, as alternativas são listadas dentro de chaves ('{' e '}'):

```
{DESCRIBE | DESC} nome_tbl {nome_col | metacar}
```

1.2 Visão Geral do Sistema de Gerenciamento de Banco de Dados MySQL

MySQL, o mais popular sistema de gerenciamento de banco de dados SQL Open Source, é desenvolvido, distribuído e tem suporte da MySQL AB. A MySQL AB é uma empresa comercial, fundada pelos desenvolvedores do MySQL, cujos negócios é fornecer serviços relacionados ao sistema de gerenciamento de banco de dados MySQL. Veja Seção 1.3 [What is MySQL AB], Página 12.

O web site do MySQL (http://www.mysql.com/) fornece informações mais recentes sobre e programa MySQL e a MySQL AB.

O MySQL é um sistema de gerenciamento de bancos de dados.

Um banco de dados é uma coleção de dados estruturados. Ele pode ser qualquer coisa desde uma simples lista de compras a uma galeria de imagens ou a grande quantidade de informação da sua rede coorporativa. Para adicionar, acessar, e processar dados armazenados em um banco de dados de um computador, você necessita de um sistema de gerenciamento de bancos de dados como o Servidor MySQL. Como os computadores são muito bons em lidar com grandes quantidades de dados, o gerenciamento de bancos de dados funciona como a engrenagem central na computação, seja como utilitários independentes ou como partes de outras aplicações.

O MySQL é um sistema de gerenciamento de bancos de dados relacional.

Um banco de dados relacional armazena dados em tabelas separadas em vez de colocar todos os dados um só local. Isso proporciona velocidade e flexibilidade. A parte SQL do "MySQL" atenda pela "Structured Query Language - Linguagem Estrutural de Consultas". SQL é linguagem padrão mais comum usada para acessar banco de dados e é definida pelo Padrão ANSI/ISO SQL. (O padrão SQL está vem evoluindo desde 1986 e existem diversas versões. Neste manual, "SQL-92" se refere ao padrão liberado em 1992, "SQL-99" se refere ao padrão liberado em 1999, e "SQL:2003" se refere a versão do que esperamos que seja liberado no meio de 2003. Nós usamos o termo "o padrão SQL" indicando a versão atual do Padrão SQL em qualquer momento).

O é MySQL um software Open Source.

Open Source significa que é possível para qualquer um usar e modificar o programa. Qualquer pessoa pode fazer download do MySQL pela Internet e usá-lo sem pagar nada. Se você quiser, você pode estudar o código fonte e alterá-lo para adequá-lo às suas necessidades. O MySQL usa a GPL (GNU General Public License - Licença Pública Geral GNU) http://www.fsf.org/licenses, para definir o que você pode e não pode fazer com o software em diferentes situações. Se você sentir desconforto com a GPL ou precisar embutir o MySQL

em uma aplicação comercial você pode adquirir a versão comercial licenciada conosco. Veja Seção 1.4.3 [Licenças MySQL], Página 18.

Por que usar o Banco de Dados MySQL?

O `servidor de banco de dados MySQL` é extremamente rápido, confiável, e fácil de usar. Se isto é o que você está procurando, você deveria experimentá-lo. O `Servidor MySQL` também tem um conjunto de recursos muito práticos desenvolvidos com a cooperação de nossos usuários. Você pode encontrar comparativos de performance do `Servidor MySQL` com outros gerenciadores de bancos de dados na nossa página de benchmark Veja Seção 5.1.4 [MySQL Benchmarks], Página 421.

O `Servidor MySQL` foi desenvolvido originalmente para lidar com bancos de dados muito grandes de maneira muito mais rápida que as soluções existentes e tem sido usado em ambientes de produção de alta demanda por diversos anos de maneira bem sucedida. Apesar de estar em constante desenvolvimento, o `Servidor MySQL` oferece hoje um rico e proveitoso conjunto de funções. A conectividade, velocidade, e segurança fazem com que o `MySQL` seja altamente adaptável para acessar bancos de dados na Internet.

As características técnicas do MySQL

Para informações técnicas avançadas, veja Cap“ptexi tulo 6 [Reference], Página 469. O `Programa de Banco de Dados MySQL` é um sistema cliente/servidor que consiste de um servidor `SQL` multi-tarefa que suporta acessos diferentes, diversos programas clientes e bibliotecas, ferramentas administrativas e diversas interfaces de programação (`API's`).

Também concedemos o `Servidor MySQL` como uma biblioteca multi-tarefa que você pode ligar à sua aplicação para chegar a um produto mais rápido, menor e mais fácilmente gerenciável.

MySQL tem muitos softwares de colaboradores disponível.

É bem provável que sua aplicação ou linguagem favorita já suporte o `Servidor de Banco de Dados MySQL`.

A pronúncia oficial do `MySQL` é “Mai Ess Que Ell” (e não MAI-SEQUEL). Mas nós não ligamos se você pronunciar MAI-SEQUEL ou de outra forma qualquer.

1.2.1 História do MySQL

Quando começamos, tínhamos a intenção de usar o `mSQL` para conectar às nossas tabelas utilizando nossas rápidas rotinas de baixo nível (ISAM). Entretanto, depois de alguns testes, chegamos a conclusão que o `mSQL` não era rápido e nem flexível o suficiente para nossas necessidades. Isto resultou em uma nova interface SQL para nosso banco de dados, mas com praticamente a mesma Interface API do `mSQL`. Esta API foi escolhida para facilitar a portabilidade para códigos de terceiros que era escrito para uso com `mSQL` para ser portado facilmente para uso com o `MySQL`.

A derivação do nome MySQL não é bem definida. Nosso diretório base e um grande número de nossas bibliotecas e ferramentas sempre tiveram o prefixo “my” por pelo menos 10 anos. A filha de Monty também ganhou o nome My. Qual das duas originou o nome do MySQL continua sendo um mistério, mesmo para nós.

O nome do golfinho do MySQL (nosso logo) é `Sakila`. `Sakila` foi escolhido pelos fundadores da MySQL AB de uma enorme lista de nomes sugeridos pelos usuários em nosso concurso "Name the Dolphin". O nome vencedor foi enviado por Ambrose Twebaze, um desenvolvedor de programas open source de Swaziland, Africa. De acordo com Ambrose, o nome Sakila tem as suas raízes em SiSwati, a língua local de Swaziland. Sakila é também o nome de uma cidade em Arusha, Tanzania, próxima ao país de orígem de Ambrose, Uganda.

1.2.2 As Principais Características do MySQL

A seguinte lista descreve algumas das características mais importantes do `Progrma de Banco de Dados MySQL`. Veja Seção 1.5.1 [MySQL 4.0 Nutshell], Página 22.

Portabilidade e

- Escrito em C e C++.
- Testado com um amplo faixa de compiladores diferentes.
- Funciona em diversas plataformas. Veja Seção 2.2.3 [Quais SO], Página 78.
- Utiliza o GNU Automake, Autoconf, e Libtool para portabilidade.
- APIs para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby e Tcl estão disponíveis. Veja Cap“ptexi tulo 12 [Clients], Página 775.
- Suporte total a multi-threads usando threads diretamente no kernel. Isto significa que se pode facilmente usar múltiplas CPUs, se disponível.
- Fornece mecanismos de armazenamento transacional e não transacional.
- Tabelas em disco (`MyISAM`) baseadas em árvores-B extremamente rápidas com compressão de índices.
- É relativamente fácil se adicionar outro mecanismo de armazenamento. Isto é útil se você quiser adicionar uma interface SQL a um banco de dados caseiro.
- Um sistema de alocação de memória muito rápido e baseado em processo(thread).
- Joins muito rápidas usando uma multi-join de leitura única otimizada.
- Tabelas hash em memória que são usadas como tabelas temporárias.
- Funções SQL são implementadas por meio de uma biblioteca de classes altamente otimizada e com o máximo de performance. Geralmente não há nenhuma alocação de memória depois da inicialização da pesquisa.
- O código do `MySQL` foi testado com Purify (um detector comercial de falhas de memória) e também com o Valgrind, uma ferramenta `GPL` (`http://developer.kde.org/~sewardj/`).
- Disponível como versão cliente/servidor ou embutida(ligada).

Tipos de Coluna

- Aceita diversos tipos de campos: tipos inteiros de 1, 2, 3, 4 e 8 bytes com e sem sinal, `FLOAT`, `DOUBLE`, `CHAR`, `VARCHAR`, `TEXT`, `BLOB`, `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, `YEAR`, `SET` e `ENUM`. Veja Seção 6.2 [Tipos de Coluna], Página 482.
- Registros de tamanhos fixos ou variáveis.

Comandos e Funções

- Completo suporte a operadores e funções nas partes `SELECT` e `WHERE` das consultas. Por exemplo:

```
mysql> SELECT CONCAT(first_name, " ", last_name)
    -> FROM nome_tbl
    -> WHERE income/dependents > 10000 AND age > 30;
```

- Suporte pleno às cláusulas SQL `GROUP BY` e `ORDER BY`. Suporte para funções de agrupamento (`COUNT()`, `COUNT(DISTINCT ...)`, `AVG()`, `STD()`, `SUM()`, `MAX()` e `MIN()`).
- Suporte para `LEFT OUTER JOIN` e `RIGHT OUTER JOIN` com as sintaxes SQL e ODBC.
- Alias em tabelas e colunas são disponíveis como definidos no padrão SQL92.
- `DELETE`, `INSERT`, `REPLACE`, e `UPDATE` retornam o número de linhas que foram alteradas (afetadas). É possível retornar o número de linhas com padrão coincidentes configurando um parâmetro quando estiver conectando ao servidor.
- O comando específico do `MySQL SHOW` pode ser usado para devolver informações sobre bancos de dados, tabelas e índices. O comando `EXPLAIN` pode ser usado para determinar como o otimizador resolve a consulta.
- Nomes de funções não conflitam com nomes de tabelas ou colunas. Por exemplo, `ABS` é um nome de campo válido. A única restrição é que para uma chamada de função, espaços não são permitidos entre o nome da função e o '(' que o segue. Veja Seção 6.1.7 [Palavras reservadas], Página 479.
- Você pode misturar tabelas de bancos de dados diferentes na mesma pesquisa (como na versão 3.22).

Segurança

- Um sistema de privilégios e senhas que é muito flexível, seguro e que permite verificação baseada em estações/máquinas. Senhas são seguras porque todo o tráfico de senhas é criptografado quando você se conecta ao servidor.

Escalabilidade e limites

- Lida com bancos de dados enormes. Usamos o `Servidor MySQL` com bancos de dados que contém 50.000.000 registros e sabemos de usuários que usam o `Servidor MySQL` com 60.000 tabelas e aproximadamente 5.000.000.000 de linhas.
- São permitidos até 32 índices por tabela. Cada índice pode ser composto de 1 a 16 colunas ou partes de colunas. O tamanho máximo do índice é de 500 bytes (isto pode ser alterado na compilação do MySQL). Um índice pode usar o prefixo de campo com um tipo `CHAR` ou `VARCHAR`.

Conectividade

- Os clientes podem se conectar ao servidor MySQL usando sockets TCP/IP, em qualquer plataforma. No sistema Windows na família NT (NT, 2000 ou XP), os clientes podem se conectar usando named pipes. No sistema Unix, os clientes podem se conectar usando arquivos sockets.

- A interface Connector/ODBC fornece ao MySQL suporte a progras clientes que usam conexão ODBC (Open-DataBase-Connectivity). Por exemplo, você pode usar o MS Access para conectar ao seu servidor `MySQL`. Os clientes podem ser executados no Windows ou Unix. O fonte do Connector/ODBC está disponível. Todas as funções ODBC são suportadas, assim como muitas outras.

  Veja Seção 12.2 [ODBC], Página 870.

Localização

- O servidor pode apresentar mensagem de erros aos clientes em várias línguas. Veja Seção 4.7.2 [Languages], Página 327.
- Suporte total para vários conjuntos de caracteres, que incluem ISO-8859-1 (Latin1), big5, ujis e mais. Por exemplo, os caracteres Escandinavos '`å`', '`ä`', '`ö`' são permitidos em nomes de tabelas e colunas.
- Todos os dados são armazenados no conjunto de caracteres escolhido. Todas as comparações em colunas de seqüências caso-insensitivo.
- A ordenação é feita de acordo com o conjunto de caracteres escolhido (o modo sueco por padrão). É possível alterar isso quando o servidor `MySQL` é iniciado. Para ver um exemplo de várias ordenações avançadas, procure pelo código de ordenação Tcheca. O `Servidor MySQL` suporta diversos conjuntos de caracteres que podem ser especificados em tempo de compilação e execução.

Clientes e Ferramentas

- O servidor MySQL foi construído com suporte para instruções SQL que verificam, otimizam e reparam tabelas. Estas instruções estão disponíveis a partir da linha de comando por meio do cliente `myisamcheck`, O MySQL inclui também o `myisamchk`, um utilitário muito rápido para realizar estas operações em tabelas `MyISAM`. Veja Cap"ptexi tulo 4 [MySQL Database Administration], Página 207.
- Todos os programas `MySQL` podem ser chamados com as opções `--help` ou `-?` para obter ajuda online.

1.2.3 Estabilidade do MySQL

Esta seção discute as questões *"Quão estável é o MySQL?"* e *"Posso depender do MySQL neste projeto?"*. Tentaremos deixar claro estes assuntos e responder algumas das questões mais importantes que dizem respeito a muito de nossos usuários. A informação nesta seção é baseada em dados colhidos da lista de discussão, que é muito ativa na identificação de problemas e assim como nos relatos de tipos de uso.

Originalmente, o código vem do início dos anos 80, fornecendo um código estável e o formato de tabelas ISAM permanece compatível com versões anteriores. Na TcX, a predecessora da `MySQLAB`, o `MySQL` vem trabalhando sem problemas em nossos projetos desde o meio de 1996. Quando o `Programa de Banco de Dados MySQL` foi disponibilizado para um público maior, nossos novos usuários rapidamente encontraram algumas partes de "código sem testes". Desde então, cada distribuição nova teve menos problemas de portabilidade (mesmo com os novos recursos implementados em cada uma destas versões)

Cada distribuição do Servidor MySQL foi sendo usado, e os problemas tem ocorrido somente quando os usuários começam a usar o código das "áreas cinzentas." Naturalmente, novos usuários não sabem o que são as áreas cinzentas; esta seção tenta indicar aquelas que são conhecidas atualmente. As descrições lidam com a Versão 3.23 e 4.0 do Servidor MySQL. Todos os erros conhecidos e relatados são corrigidos na última versão, com a exceção dos bugs listados na seção de erros, os quais são relacionados ao desenho. Veja Seção 1.8.6 [Bugs], Página 54.

O Servidor MySQL é escrito em múltiplas camadas com módulos independentes. Alguns dos novos módulos estão listados abaixo com indicações de quão bem-testado foi cada um deles.

Replicação — Gamma
: Grandes grupos de servidores usando replicação estão em uso, com bom resultados. O trabalho no aprimoramento dos recursos de replicação continua no MySQL 4.x.

Tabelas InnoDB — Estável (na 3.23, 3.23.49)
: O mecanismo de armazenamento transacional InnoDB foi declarado estável na árvore do MySQL 3.23, a partir da versão 3.23.49. InnoDB tem sido usado em sistema de produção grandes e com carga pesada.

Tabelas BDB — Gamma
: O código do Berkeley DB é muito estável, mas ainda estamos melhorando a interface do mecanismo de armazenamento transacional do BDB no Servidor MySQL, assim levará algum tempo até que ele esteja tão bem testado quanto os outro tipos de tabela.

Pesquisas Full-text — Beta
: Pesquisa full-text funcionam mas ainda não são largamente usadas. Melhoramentos importantes forma implementados no MySQL 4.0.

MyODBC 3.51 (usa ODBC SDK 3.51) — Estável
: Em grande uso na produção. Alguns problemas apresentados parecem ser relacionados a aplicação e independente do driver ODBC ou do servidor de banco de dados.

Recuperação automática de tabelas MyISAM — Gamma
: Este status se aplica apenas ao novo código que confere no mecanismo de armazenamento MyISAM que verifica, na inicialização, se a tabela foi fechada corretamente e executa uma conferência/reparo automático da tabela em caso negativo.

Bulk-insert — Alpha
: Novo recurso nas tabelas MyISAM no MySQL 4.0 para inserções mais rápidas de vários registros.

Locking — Gamma
: Esse módulo é muito dependente do sistema. Em alguns sistemas existem certos problemas por utilizar o locking padrão do SO (`fcntl()`. Nestes casos, você deve executar o `mysqld` com o parâmetro `--skip-external-locking`. São conhecidos alguns problemas ocorridos em alguns sistemas Linux e no SunOS quando utiliza-se sistemas de arquivos montados em NFS.

Clientes que pagam recebem suporte direto e de alta qualidade da MySQL AB. A MySQL AB também fornece uma lista de discussão como um recurso da comunidade onde qualquer pessoa pode tirar suas dúvidas.

Erros são normalmente corrigidos com um patch; para erros sérios, normalmente é lançada uma nova distribuição.

1.2.4 Qual o Tamanho Que as Tabelas do MySQL Podem Ter?

A Versão 3.22 do `MySQL` tem suporte para tabelas com limite de tamanho até 4G. Com o novo `MyISAM` no `MySQL` versão 3.23 o tamanho máximo foi expandido até 8 milhões de terabytes (2 ^ 63 bytes). Com este tamanho de tabela maior permitido, o tamanho máximo efetivo das tabelas para o banco de dados `MySQL` é normalmente limitado pelas restrições do sistema operacional quanto ao tamanho dos arquivos, não mais por limites internos do MySQL.

A seguinte tabela lista alguns exemplos do limite do tamanho de arquivos do sistema operacional:

Sistema Operacional	**Limite do tamanho do arquivo**
Linux-Intel 32 bit	2G, muito mais usando LFS
Linux-Alpha	8T (?)
Solaris 2.5.1	2G (É possível 4GB com patch)
Solaris 2.6	4G (pode ser alterado com parâmetro)
Solaris 2.7 Intel	4G
Solaris 2.7 ULTRA-SPARC	8T (?)

No Linux 2.2 você pode ter tabelas maiores que 2 GB usando o patch LFS para o sistema de arquivos ext2. No Linux 2.4 já existem patches para o sistema de arquivos ReiserFS para ter suporte a arquivos maiores. A maioria das distribuições atuais são baseadas no kernel 2.4 e já incluem todos os patches Suporte a Arquivos Grandes (Large File Support - LFS) exigidos. No entanto, o tamanho máximo disponível ainda depende de diversos fatores, sendo um deles o sistema de arquivos usado para armazenar as tabelas MySQL.

Para um visão mais detalhada sobre LFS no Linux, dê uma olha na página Andreas Jaeger's "Large File Support in Linux" em `http://www.suse.de/~aj/linux_lfs.html`.

Por padrão, o `MySQL` cria tabelas `MyISAM` com uma estrutura interna que permite um tamanho máximo em torno de 4G. Você pode verificar o tamanho máximo da tabela com o comando `SHOW TABLE STATUS` ou com o `myisamchk -dv nome_tabela` Veja Seção 4.6.8 [`SHOW`], Página 302.

Se você precisa de tabelas maiores que 4G (e seu sistema operacional suporta arquivos grandes), a instrução `CREATE TABLE` permite as opções `AVG_ROW_LENGHT` e `MAX_ROWS`. Use estas opções para criar uma tabela que possa ter mais de 4GB. Veja Seção 6.5.3 [`CREATE TABLE`], Página 598. Você pode também alterar isso mais tarde com `ALTER TABLE`. Veja Seção 6.5.4 [`ALTER TABLE`], Página 608.

Outros modos se contornar o limite do tamanho do arquivo das tabelas `MyISAM` são os seguintes:

- Se sua tabela grande será somente leitura, você poderá usar o `myisampack` para unir e comprimir várias tabelas em uma. `mysisampack` normalmente comprime uma tabela

em pelo menos 50%, portanto você pode obter, com isso, tabelas muito maiores. Veja Seção 4.8.4 [myisampack], Página 336.

- Outra opção para contornar o limite de tamanho de arquivos do sistema operacional para arquivos de dados MyISAM usando a opção RAID. Veja Seção 6.5.3 [CREATE TABLE], Página 598.
- O MySQL incluí uma biblioteca MERGE que permite acessar uma coleção de tabelas idênticas como se fosse apenas uma. Veja Seção 7.2 [MERGE], Página 638.

1.2.5 Compatibilidade Com o Ano 2000 (Y2K)

O Servidor MySQL não apresenta nenhum problema com o ano 2000 (Y2K compatível)

- O Servidor MySQL usa funções de tempo Unix que tratam datas até o ano 2037 para valores TIMESTAMP; para valores DATE e DATETIME, datas até o ano 9999 são aceitas.
- Todas as funções de data do MySQL estão no arquivo 'sql/time.cc' e codificadas com muito cuidado para ser compatível com o ano 2000.
- No MySQL versão 3.22 e posterior, o novo tipo de campo YEAR pode armazenar anos 0 e 1901 até 2155 em 1 byte e mostrá-lo usando 2 ou 4 dígitos. Todos os anos de 2 dígitos são considerados estar na faixa de 1970 até 2069; o que significa que se você armazenar 01 em uma coluna YEAR, O Servidor MySQL o tratará como 2001.

O seguinte demonstração simples ilustra que o MySQL Server não tem nenhum problema com datas até depois do ano 2030:

```
mysql> DROP TABLE IF EXISTS y2k;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE y2k (date DATE,
    ->                   date_time DATETIME,
    ->                   time_stamp TIMESTAMP);
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO y2k VALUES
    -> ("1998-12-31","1998-12-31 23:59:59",19981231235959),
    -> ("1999-01-01","1999-01-01 00:00:00",19990101000000),
    -> ("1999-09-09","1999-09-09 23:59:59",19990909235959),
    -> ("2000-01-01","2000-01-01 00:00:00",20000101000000),
    -> ("2000-02-28","2000-02-28 00:00:00",20000228000000),
    -> ("2000-02-29","2000-02-29 00:00:00",20000229000000),
    -> ("2000-03-01","2000-03-01 00:00:00",20000301000000),
    -> ("2000-12-31","2000-12-31 23:59:59",20001231235959),
    -> ("2001-01-01","2001-01-01 00:00:00",20010101000000),
    -> ("2004-12-31","2004-12-31 23:59:59",20041231235959),
    -> ("2005-01-01","2005-01-01 00:00:00",20050101000000),
    -> ("2030-01-01","2030-01-01 00:00:00",20300101000000),
    -> ("2050-01-01","2050-01-01 00:00:00",20500101000000);
Query OK, 13 rows affected (0.01 sec)
Records: 13  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM y2k;
+------------+---------------------+----------------+
| date       | date_time           | time_stamp     |
+------------+---------------------+----------------+
| 1998-12-31 | 1998-12-31 23:59:59 | 19981231235959 |
| 1999-01-01 | 1999-01-01 00:00:00 | 19990101000000 |
| 1999-09-09 | 1999-09-09 23:59:59 | 19990909235959 |
| 2000-01-01 | 2000-01-01 00:00:00 | 20000101000000 |
| 2000-02-28 | 2000-02-28 00:00:00 | 20000228000000 |
| 2000-02-29 | 2000-02-29 00:00:00 | 20000229000000 |
| 2000-03-01 | 2000-03-01 00:00:00 | 20000301000000 |
| 2000-12-31 | 2000-12-31 23:59:59 | 20001231235959 |
| 2001-01-01 | 2001-01-01 00:00:00 | 20010101000000 |
| 2004-12-31 | 2004-12-31 23:59:59 | 20041231235959 |
| 2005-01-01 | 2005-01-01 00:00:00 | 20050101000000 |
| 2030-01-01 | 2030-01-01 00:00:00 | 20300101000000 |
| 2050-01-01 | 2050-01-01 00:00:00 | 00000000000000 |
+------------+---------------------+----------------+
13 rows in set (0.00 sec)
```

O valor da coluna `TIMESTAMP` final é zero porque o ano final (`2050`) excede o `TIMESTAMP` maximo. O tipo de dados `TIMESTAMP`, que é usado para armazenar a hora atual, suporta valores na faixa de `19700101000000` a `20300101000000` em máquinas 32 bits (valor com sinal). Em máquinas de 64 bits, `TIMESTAMP` trata valores até `2106` (valores sem sinal).

O exemplo mostra que os tipos `DATE` e `DATETIME` não tem problemas com as datas usadas. Eles irão conseguir trabalhar com datas até o ano `9999`.

Embora o `MySQL Server` seja seguro em relação ao ano 2000, você pode ter problemas se você usá-lo com aplicações que não são seguras com o ano 2000. Por exemplo, muitas aplicações antigas armazenam ou manipulam anos usando valores de 2 digitos (que são ambíguos) em vez de 4 dígitos. Este problema pode ser aumentado por aplicações que usam valores como `00` ou `99` como indicadores de valores "perdidos". Infelizmente, estes problemas pode ser difíceis de corrigir, cada um deles pode usar um conjunto diferente de convenções e funções de tratamento de datas.

Assim, apesar do `Servidor MySQL` não ter problemas com o ano 2000, é de responsabilidade de sua aplicação fornecer datas que não sejam ambíguas. Veja Seção 6.2.2.1 [Y2K issues], Página 490 para as regras do `Servidor MySQL` para lidar com entrada de datas ambíguas que contenham valores de ano com 2 dígitos.

1.3 Visão Geral da MySQL AB

`MySQL AB` é a companhia dos fundadores e principais desenvolvedores do `MySQL`. A `MySQL AB` foi estabelecida originalmente na Suécia por David Axmark, Allan Larsson e Michael "Monty" Widenius.

Os desenvolvedores do servidor MySQL são todos empregados pela companhia. ny Nós somo uma organização virtual com pessoas em uma dúzia de países. Nos comunicamos extensivamente na internet todos os dias uns com os outros e com nossos usuários, agentes de suporte e parceiros.

Nós nos dedicamos a desenvolver o programa MySQL e propagar nosso banco de dados a novos usuários.A MySQL AB detém os direitos autorais do código fonte do MySQL, do logo e da marca MySQL e deste manual. Veja Seção 1.2 [What-is], Página 4.

A ideologia do MySQL mostra nossa dedicação ao MySQL e ao Open Source.

Nós desejamos que o Programa de Banco de Dados MySQL seja:

O melhor e o mais usado banco de dados no mundo.

- Acessível e disponível para todos.
- Fácil de usar.
- Melhorado continuamente, permanecendo rápido e seguro.
- Divertido de se usar e aprimorar.
- Livre de erros (bugs).

A MySQL AB e sua equipe:

- Promovem a filosofia Open Source e suporte à comunidade Open Source.
- Tem como objetivo serem bons cidadãos.
- Tem preferência por parceiros que compartilhem nossos valores e idéias.
- Respondem e-mails e dão suporte.
- São uma empresa virtual, conectada com outras.
- Trabalha contra patentes de sistemas.

O site do MySQL (http://www.mysql.com/) fornece as últimas informações sobre o MySQL e a MySQL AB.

A propósito, a parte "AB" do nome da companhia é o acrônimo para a palavra suéca "aktiebolag", ou "sociedade anônima." Ela é traduzida para "MySQL, Inc." De fato, MySQL Inc. e MySQL GmbH são exemplos de subsidiárias da MySQL AB. Elas estão localizadas nos EUA e Alemanha, respectivamente.

1.3.1 O Modelo de Negócio e Serviços da MySQL AB

Uma das dúvidas mais comuns que encontramos é: "*Como você pode viver de algo que você disponibiliza sem custo?*" É assim que fazemos.

A MySQL AB ganha dinheiro com suporte, serviços, licenças comerciais e royalties. Usamos estes rendimentos para patrocinar o desenvolvimento de produtos e para expandir os negócios da MySQL.

A compania tem sido luccrativa desde de sua criação. Em Outubro de 2001, aceitamos um financiamento de risco de investidores Escandinavos e um pounhado de pessoas de negócio. Este investimento é usado para solidificarmos nosso modelo de negócio e construir um base para o crescimento sustentável.

1.3.1.1 Suporte

A MySQL AB é gerenciada pelos fundadores e principais desenvolvedores do banco de dados MySQL. Os desenvolvedores tem o compromisso de dar suporte aos clientes e outros usuários com objetivo de manterem contato com as suas necessiades e problemas. Todo o nosso suporte é dado por desenvolvedores qualificado. Dúvidas mais complicadas são respondidas por Michael Monty Widenius, principal autor do MySQL Server. Veja Seção 1.4.1 [Suporte], Página 17.

Para maiores informações e pedido de suporte de diversos níveis, veja http://www.mysql.com/support/ ou contate nossos vendedores em sales@mysql.com.

1.3.1.2 Treinamento e Certificação

A MySQL AB distribui o MySQL e treinamentos relacionados mundialmente. Oferecemos tanto cursos abertos quanto fechados voltado para a necessidade específica da sua empresa. O Treinamento do MySQL também está disponível por meio de seus parceiros, os Centros de Treinamento Autorizados do MySQL.

Nosso material de treinamento usa os mesmos bancos de dados exemplos usados em nossa documentação e nossos exemplos de aplicativos. Ele está sempre atualizado de acordo com a última versão do MySQL. Nossos instrutores são apoiados por nossa equipe de desenvolvimento para garantir a qualidade do treinamento e o desenvolvimento contínuo do material de nossos cursos. Isto também assegura que nenhuma questão surgida durante o curso fique sem resposta.

Fazendo nossos cursos de treinamento permitirá que você alcance os objetivos de seu aplicativo MySQL. você também irá:

- Economizar tempo.
- Melhorar o desempenho de seus aplicativos.
- Reduzir ou eliminar a necessidade de hardware adicional, baixando o custo.
- Melhorar a segurança.
- Aumentar a satisfação dos clientes e colabloradores.
- Preparar-se para Certificação MySQL.

Se você estiver interessado em nosso treinamento como um participante em portencial ou como um parceiro de treinamento, viste a seção de treinamento em http://www.mysql.com/training/ ou contate nos em: training@mysql.com.

Para detalhes sobre o Programa de Certificação MySQL, veja http://www.mysql.com/certification/.

1.3.1.3 Consultoria

A MySQL AB e seus Parceiros Autorizados oferecem serviços de consultoria para usuários do Servidor MySQL e àqueles que utilizam o Servisdor MySQL embutido em seus programas, em qualquer parte do mundo.

Nossos consultores podem ajudá-lo projetando e ajustando o seu banco de dados, criar consultas eficientes, ajustar sua plataforma para uma melhor performance, resolver questões

de migração, configurar replicação, criar aplicações transacionais robustas, e mais. Também ajudamos os nossos clientes com o Servidor MySQL embutido em seus produtos e aplicações para desenvolvimento em larga-escala.

Nossos consultores trabalham em colaboração com a nossa equipe de desenvolvimento, o que assegura a qualidade técnica de nossos serviços profissionais. Os serviços de consultoria varia de sessões de 2 dias a projetos que gastam semanas e meses. Nosso peritos não apenas cobrem o Servidor MySQLeles também conhecem sobre linguagens de programação e scripts tais como PHP, Perl e mais.

Se estiver interessado em nossos serviços de consultoria ou quiser se tornar nosso parceiro, visite a seção sobre consultaria em nosso web site em http://www.mysql.com/consulting/ ou contate nossa equipe de consultoria em consulting@mysql.com.

1.3.1.4 Licenças Comerciais

O banco de dados MySQL é liberado sob a licença GNU General Public License (GPL). Isto significa que o programa MySQL pode ser usado sem custos sob a GPL. Se você não deseja estar limitado pelos termos da GPL (tais como a exigência de que a sua aplicação também deva ser GPL), você pode comprar uma licença comercial para o mesmo produto da MySQL AB; veja http://www.mysql.com/products/pricing.html. Desde de que a MySQL AB é dona dos direitos do código fonte do MySQL, estamos aptos a empregar o Licenciamento Dual, que significa que o mesmo produto está disponível sob a GPL e sob uma licença comercial. Isto não afeta o nosso comprometimento com o Open Source de forma alguma. Para detalhes sobre quando uma licença comercial é exigida, veja Seção 1.4.3 [MySQL licenses], Página 18.

1.3.1.5 Parcerias

A MySQL AB tem um programa de parceria mundial que cobre cursos de treinamento, consultaria e suporte, publicações, mais a revenda e distribiuição do MySQL e produtos relacionados. Os Parceiros da MySQL AB ganham visibilidade no nosso web site (http://www.mysql.com/) e o direito de usarem versões especiais da marca MySQL para identificar seus produtos e promoverem os seus negócios.

Se você está interessado em se tornar um Parceiro da MySQL AB, envie-nos um email para partner@mysql.com.

A palavra MySQL e o logomarca do golfinho da MySQL são marcas registradas da MySQL AB. Veja Seção 1.4.4 [MySQL AB Logos and Trademarks], Página 20. Estas marcas registradas representam um valor significante que os fundadores do MySQL construiram ao longo dos anos.

O web site do MySQL (http://www.mysql.com/) é popular entre desenvolvedores e usuários. Em Outubro de 2001, obtivemos mais de 10 milhões e views. Nossos visitantes representam um grupo que tomam decisões de compra e fazem recomendções de software e hardware. Vinte por cento de nossos vistantes autorizam decisões de compra e apenas nove por cento não estão envolvidos com a área de compras. Mais de 65% fizeram uma ou mais compras online no último semaster e 70% planejam fazer uma compra nos próximos meses.

1.3.2 Informações para Contato

O web site do MySQL (http://www.mysql.com/) fornece as últimas informações sobre MySQL e MySQL AB.

Para serviços de imprensa e questões não cobertas por nossas releases de nottícias (http://www.mysql.com/news/), envie-nos um email para press@mysql.com.

Se você tiver um contrato de suporte válido com a MySQL AB, você receberá em tempo, respostas precisas para as suas questões técnicas sobre o programa MySQL. Para mais informações, veja Seção 1.4.1 [Support], Página 17. Em nosso site na web, veja http://www.mysql.com/support/, ou envie um e-mail para sales@mysql.com.

Para informações sobre treinamento MySQL, visite a seção de treinamento em http://www.mysql.com/training/. Se você tiver acesso restrito à Internet, conte a equipe de treinamento da MySQL AB via e-mail em training@mysql.com. Veja Seção 1.3.1.2 [Business Services Training], Página 14.

Para informações sobre o Progrma de Certificaço MySQL, veja http://www.mysql.com/certification/. Veja Seção 1.3.1.2 [Business Services Training], Página 14.

Se você estiver interessado em consultoria, visite a seção de consultorias de nosso web site em http://www.mysql.com/consulting/. Se você tiver restrições acesso a internet, contate a equipe de consultores da MySQL AB via e-mail em consulting@mysql.com. Veja Seção 1.3.1.3 [Business Services Consulting], Página 14.

Licenças comerciais podem ser compradas online em https://order.mysql.com/. Lá você também encontrará informações de como enviar um fax da sua ordem de compra para a MySQL AB. Mais informações sobre licenças podem ser encontradas em http://www.mysql.com/products/pricing.html. Se você tiver duvidas em relação a licenciamento ou quiser cota para negociação de um alto volume de licenças, preencha o formulário de contato em nosso site web (http://www.mysql.com/) ou envie um email para licensing@mysql.com (para questões sobre licenciamento) ou para sales@mysql.com (para pedidos de compra). Veja Seção 1.4.3 [Licenças MySQL], Página 18.

Se você está interessado em fazer parceira com a MySQL AB, envie um e-mail para partner@mysql.com. Veja Seção 1.3.1.5 [Business Services Partnering], Página 15.

Para mais detalhes sobre a polítca da marca MySQL, visite http://www.mysql.com/company/trademark.ht ou envie um e-mail para trademark@mysql.com. Veja Seção 1.4.4 [MySQL AB Logos and Trademarks], Página 20.

Se você está interessado em qualquer um dos trabalhos da MySQL AB lista na seção de trabalhos (http://www.mysql.com/company/jobs/), envie um e-mail para jobs@mysql.com. Não nos envie o seu CV em anexo, mas como texto no final de sua mensagem de email.

Para discussões gerais entre nosso muitos usuários, direcione a sua atenção para a lista de discussão apropriada. Veja Seção 1.7.1 [Dúvidas], Página 33.

Relatórios de erros (geralmente chamados bugs), assim como questões e comentários, devem ser enviados para a lista de email geral do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33. Caso você encontre um bug de segurança importante no MySQL Server, envie-nos um e-mail para security@mysql.com. Veja Seção 1.7.1.3 [Bug reports], Página 36.

Se você tiver resultados de benchmarks que podemos publicar, contate-nos via e-mail em benchmarks@mysql.com.

Se você tiver sugestões a respeito de adições ou conexões para este manual, envie-os para a equipe do manual via e-mail em `docs@mysql.com`.

Para questões ou comentários sobre o funcionamento ou coteúdo do web site da `MySQL` (`http://www.mysql.com/`), envie um e-mail para `webmaster@mysql.com`.

A `MySQL AB` tem uma política de privacidade que pode ser lida em `http://www.mysql.com/company/privac` Para qualquer questões a respeito desta política, envie um e-mail para `privacy@mysql.com`.

Para todos os outros assunto, envie um e-mail para `info@mysql.com`.

1.4 Suporte e Licenciamento do MySQL

Esta seção descreve os contratos de licenciamento e suporte do `MySQL`.

1.4.1 Suporte Oferecido pela MySQL AB

O suporte técnico do `MySQL AB` significa respostas individualizadas as seus problemas particulares diretamente dos engenheiros de software que codificaram o `MySQL`.

Tentamos ter uma visão ampla e inclusiva de suporte técnico. Qualquer problema envolvendo o `MySQL` é importante par nós se for importante para você. Normalmente os clientes procuram ajuda em como comandos e utilitários diferentes funcionam, remover gargalos de desempenhos, restaurar sistemas com falhas, entender impactos do sistema operacional e rede no `MySQL`, configurar melhor práticas de backup e restauração, utiluizaar `API`s, e assim por diante. Nosso suporte cobre apenar o servidor `MySQL` e nossos próprios utilitários, e não produtos de terceirosque acessam o servidor `MySQL`, embora tentamos ajudar com eles quando podemos.

Informações detalhadas sobre nossas várias opções de suporte é dado em

Suporte técnico é como seguro de vida. Você pode viver felizsem ele durante anos, mas quando sua hora chegar ele é de grande importância, mas já é muito tarde para adquiri-lo. Se você utiliza o `MySQL Server` para aplicações importantes e encontrar dificuldades repentinas, você pode gastar horas tentando resolver os problemas sozinho. Você pode precisar de acesso imediato aos responsáveis pela solução de problemas do `MySQL` dsiponíveis, contratados pela `MySQL AB`.

1.4.2 Copyrights e Licenças Usadas pelo MySQL

`MySQL AB` possui os direitos sobre o código fonte do `MySQL`, as logomarcas e marcas registradas do `MySQL` e este manual. Veja Seção 1.3 [What is MySQL AB], Página 12. Diversas licenças são relevantes a distribuição do `MySQL`:

1. Todo o código específico do `MySQL` no servidor, a biblioteca `mysqlclient` e o cliente, assim como a biblioteca `GNU readline` é coberta pela `GNU General Public License`. Veja Apêndice H [GPL license], Página 1096. O texto desta licença podee ser encontrado no arquivo '`COPYING`' na distribuição.
2. A biblioteca `GNU getopt` é coberta pela `GNU Lesser General Public License`. Veja `http://www.fsf.org/licenses/`.

3. Algumas partes da fonte (a biblioteca `regexp`) é coberta por um copyright no estilo Berkeley.
4. Versões mais antiga do `MySQL` (3.22 e anteriror) estão sujeitos a uma licença estrita (`http://www.mysql.com/products/mypl.html`). Veja a documentação da versão específica para mais informação.
5. O manual de referência do `MySQL` atualmente **não** é distribuído sob uma licecnça no estilo da `GPL`. O uso deste manual está sujeito aos seguintes termos:
   - A conversão para outros formatos é permitido, mas o conteúdo atual não pode ser alterado ou editado de modo algum.
   - Você pode criar uma cópia impressa para seu próprio uso pessoal.
   - Para todos os outros usos, como venda de cópias impressas ou uso (de partes) do manual em outra publicação, é necessários um acordo com a `MySQL AB` previamente escrito.

   Envie-nos email para `docs@mysql.com` para maiores informações ou se você estiver interessado em fazer a tradução.

Para informações sobre como as licenças do `MySQL` funcionam na prática, de uma olhada em Seção 1.4.3 [MySQL licenses], Página 18. Veja também Seção 1.4.4 [MySQL AB Logos and Trademarks], Página 20.

1.4.3 Licenças do MySQL

O programa `MySQL` é distribuído sob a `GNU General Public License` (`GPL`), que é provavelmente a melhor licença `Open Source` conhecida. Os termos formais da licença `GPL` pode ser encontrado em `http://www.fsf.org/licenses/`. Veja também `http://www.fsf.org/licenses/gpl-faq.html` e `http://www.gnu.org/philosophy/enforcing-gpl.htm`

Como o programa `MySQL` é distribuído sob a `GPL`, ele pode ser usa geralmente de graça, mas para certos usos você pode querer ou precisar comprar lincenças comerciais da `MySQL AB` em `https://order.mysql.com/`. Veja `http://www.mysql.com/products/licensing.html` para mais informações.

Versões mais antigas do `MySQL` (3.22 e anteriores) estão sujeitos a uma licença mais estrita (`http://www.mysql.com/products/mypl.html`). Veja a documentação da versão específica para mais informação.

Note que o uso do programa `MySQL` sob uma licença comercial, `GPL` ou a antiga licença do `MySQL` não dá automaticamente o direito de usar as marcas registradas da `MySQL AB`. Veja Seção 1.4.4 [MySQL AB Logos and Trademarks], Página 20.

1.4.3.1 Usando o Programa MySQL Sob uma Licença Comercial

A licença `GPL` é contagiosa no sentido de que quando um programa é ligado a um programa `GPL`, todo o código fonte para todas as partes do produto resultante também devem ser distribuídas sob a `GPL`. Se você não seguir esta exigência do `GPL`, você quebra os termos da licença e perde o seu direito de usar o programa `GPL` incluído. Você também corre riscos.

Você precisará de uma licença comercial:

- Quando você liga um programa com qualquer código GPL do programa MySQL e não que que o produto resultante seja licenciado sob a GPL, talvez porque você queira criar um produto comercial ou manter fechado o código não GPL adicionado por outras razões. Ao comprar a lincença comercial, você não está usando o programa MySQL sob GPL embora o código seja o mesmo.
- Quando você distribui uma aplicação não GPL que **só** funciona com o programa MySQL e a entrega com o programa MySQL. Este tipo de solução é considerada mesmo se feita em uma rede.
- Quando você distribui cópias do programa MySQL sem fornecer o código fonte como exigido sob a licença GPL.
- Quando você quiser dar suporte adional ao desenvolvimento do banco de dados do MySQL mesmo se você não precisar formalmente de uma licença comercial. Comprar o suporte diretamente da MySQL AB é outro bom modo de contribuir com o desenvolvimento do programa MySQL, com vantagens imediatas para você. Veja Seção 1.4.1 [Support], Página 17.

Se você requisita uma licecnça, você precisará de uma para cada instalação do programa MySQL. Ela cobre qualquer número de CPUs na máquina, e nãp há nenhum limite artificial no número de clientes que conectam aom servidor de qualquer modo.

Para licenças comercias, ,visite o nosso site web em http://www.mysql.com/products/licensing.html. Para contrato de suporte, veja http://www.mysql.com/support/. Se você tiver necessidades especiais ou tiver acesso restrito a Internet, contate a nossa quipe de vendas via email em sales@mysql.com.

1.4.3.2 Usando o Programa MySQL Sem Custo Sob GPL

Você pode utilizar o programa MySQL sem custo sob a GPL se você concordar as condições do GPL. Para detalhes adicionais, incluindo respostas a duvidas comuns sobre a GPL, veja o FAQ gencio da Free Software Foundation em http://www.fsf.org/licenses/gpl-faq.html. Usos comuns da GPL incluem:

- Quando você distribui sua própria aplicação e o código fonte da MySQL com o seu produto.
- Quando você distribui o código fonte do MySQL junto com outros programas que não são ligados ou dependentes do sistema do MySQL para suas funcionalidades mesmo se você vender a distribuição comercialmente. Isto é chamado agregação na licença GPL.
- Quando você não está distribuindo **qualquer** parte do sistema do MySQL, você pode usá-lo de graça.
- Quando você for um Provedos de Serviços de Internet (Internet Service Provider - ISP), oferecendo hospedagem web com serviodres MySQL para seus clientes. Encorajamos as pessoas a usarem provedroes que possuem suporte ao MySQL, já que isto lhes dará a confiança qie seus provedores terão, de fato, os recursos para resolver qualquer problema que eles possam experimentar com a instalação do MySQL. Mesmo se um provedor não tiver uma licença comercial ara o MySQL Server, seus clientes devem ter acesso de leitura ao fonte da instalação do MySQL para que seus clientes verifiquem que ela está correta.

- Quando você usa o banco de dados MySQL em conjunto com um servidor web, você não precisa de uma licença comercial (uma vez que este não é um produto distribuído por você). Isto é verdade mesmo se você executar um servidor web comercial que utilize MySQL Server, pois você não está distribuindo qualquer parte do sistema MySQL. No entanto, neste caso nós gostariamos que você adquirisse o suporte ao MySQL pois o MySQL está ajudandoa sua empresa.

Se o seu uso do banco de dados MySQL não exige uma licença comercial, lhe encorajamos a adquirir um suporte da MySQL AB de qualquer forma. Deste modo você contribui com o desenvolvimento do MySQL e também ganha vantegens imediatas. Veja Seção 1.4.1 [Support], Página 17.

Se você utiliza o bancdo de dados MySQL em um contexto comercial e obtem lucro com o seu uso, pedimos que você ajude no desenvolvimento do MySQL adquirindo algum nível de suporte. Sentimos que se banco de dados MySQL ajudou os seus negócios, é razoável pedirmos que você ajude a MySQL AB. (De outra forma, se você nos pedir suporte, você não só estará usando de graça algo em que colocamos muito trabalhom mas também pedindo que lhe forneçamos suporte de graça também).

1.4.4 Logomarcas e Marcas Registradas da MySQL AB

Muitos usuários do banco de dados MySQL deseja mostar o logo do golfinho da MySQL AB em seus web sites,livros ou produtos fechados. Isto é bem vindo, mas deve haver anotações indicando que a palavra MySQL e o logo do golfinho da MySQL são marcas registradas da MySQL AB e só podem ser usadas como indicado na nossa política de marcas registradas em http://www.mysql.com/company/trademark.html.

1.4.4.1 O Logo Original do MySQL

O logo do golfinho do MySQL foi desenhado pela Finnish advertising agency Priority em 2001. O golfinho foi escolhido como um símbolo para o baco de dados MySQL já que ele é esperto, rápido e um animal ágil, se esforándo em navegar pelos oceanos de dados. Nós também gostamos de golfinos.

O logo original MySQL só podem ser usados pr representates da MySQL AB e aqueles que possuem um acordo escrito permitndo-os de fazê-lo.

1.4.4.2 Logomarcas da MySQL que Podem Ser Usadas Sem Permissão de Alteração

Projetamos um conjunto de logos especiais de *Uso Condicionale* que podem se encontrados em nosso site web em http://www.mysql.com/press/logos.html e usado em sites web de terceiros sem permissões de escrita da MySQL AB. O uso destas logomarcas não são totalmente irrestritas mas, como o nome indica, sujeitos a nossa política de marcas registradasque também está disponível em nosso site. Você deve ler a política de marcas registradas se plabeja usá-las. As exigências são basicamente as apresentadas aqui:

- Use a logomarca que você preciisa como mostrado no site http://www.mysql.com/. Você pode mudar sua escala para servir as suas necessidades, mas não pode alterar cores ou o desenho, ou alterar os graficos de forma alguma.

- Deixe evidente que você, e não a MySQL AB, é o criado e proprietário do site que mostra a logomarca do MySQL.
- Não use a logomarca em detrimento à MySQL AB ou ao valor das marcas registradas da MySQL AB. Nos reservamos o direito de revogar o diretiro de uso da marcas registradas da MySQL AB.
- Se você utilizar as maracas em um site da web, faça com que ele contenha um link para http://www.mysql.com/.
- Se você utilizar o banco de dados MySQL sob GPL em uma aplicação, sua aplicação deve ser Open Source deve estar apta a conectar a um servidor MySQL.

Contate-nos via e-mail em trademark@mysql.com para saber sobre os nosso acordos especiais que sirvam as suas necessidades.

1.4.4.3 Quando Você Precisa de Permissão de Alteração para Usar as Logomarcas do MySQL?

Você precisa de permissão escrita da MySQL AB antes de usar as logomarcas do MySQL nos seguintes casos:

- Quando exibir qualquer logomarca da MySQL AB em qualquer lugar diferente so seu site.
- Quando exibir qualquer logomarca da MySQL AB exceta as de *Uso Condicional* mencionadas anteiormente em sites ou outro lugar.

Devido a razões comerciais e legais monitoramos o so das marcas registradas do MySQL em proutos, livros e outros itens. Normalmente exigimos um valor para a exibição das logomarcas da MySQL AB em produtos comerciais, já que achamos razoável que parte dos rendimentos seja retornado para financiar o desenvolvimento do banco de dados MySQL.

1.4.4.4 Logomarcas dos Parceiros da MySQL AB

As logomarcas de parceria do MySQL podem ser usados apenas por companhias e pessoas que possuem um acordo de parceria por escrito com a MySQL AB. Parceiras incluem certificação com treinador ou consultor do MySQL. Para mais informações, Seção 1.3.1.5 [Partnering], Página 15.

1.4.4.5 Usando a Palavra MySQL em Texto Impresso ou Apresentação

A MySQL AB considera bem vindas as referências ao banco de dados MySQL mas deve ser indicado que a palavra MySQL é uma marca registrada da MySQL AB. Por isto, você deve adicionar o simbolo de marca registrada (TM) ao primeiro ou mais proeminente uso da palavra MySQL em um texto e, onde apropriadom indicar que MySQL é uma marca registrada da MySQL AB. Para mais informações, veja nossa política de marcas registradas em http://www.mysql.com/company/trademark.html.

1.4.4.6 Usando a Palavra `MySQL` em Nomes de Companhias e Produtos

O uso da palavra `MySQL` em nomes de produtos ou companias ou em dominios de Internet não é permitida sem permissão escrita da `MySQL AB`.

1.5 Mapa de Desenvolvimento do MySQL

Esta seção fornece uma amostra do mapa de desenvolvimento do MySQL, incluindo principais recursos imlementados ou planejados para o MySQL 4.0, 4.1, 5.0 e 5.1. A seguinte seção fornece informação para cada distribuição. O planejamento para alguns dos recursos mais requisitados estão listada na tabela a seguir.

Feature	**MySQL version**
Unions	4.0
Subqueries	4.1
R-trees	4.1 (para tabelas MyISAM)
Stored procedures	5.0
Views	5.0 ou 5.1
Cursors	5.0
Foreign keys	5.1 (3.23 com InnoDB)
Triggers	5.1
Full outer join	5.1
Constraints	5.1

1.5.1 MySQL 4.0 in a Nutshell

Muito aguardado por nossos usuários, o MySQL Server 4.0 agora está disponível em sua versão de produção.

O MySQL 4.0 está disponível para download em `http://www.mysql.com/` e nossos sites mirrors. O MySQL tem sido testado por um grande número de usuários e está em uso em mutios sites.

Os principais novos recursos do MySQL Server 4.0 são trabalhados em conjunto com os usuários corporativos e da comunidade, melhorando o programa de banco de dados MySQL como uma solução para missões críticas e sistemas de bancos de dados de alta carga. Outros novos recursos visam os usuários de bancos de dados embutidos.

O MySQL 4.0 foi declarado estável para uso em produção a partir da versão 4.0.12 em Março de 2003. Isto significa que, no futuro, apenas correção de erros serão feitas para a distribuição da série 4.0 e apenas correção de erros críticos serão feitas para a antiga série 3.23. Veja Seção 2.5.2 [Upgrading-from-3.23], Página 123.

Novos recursos para o `MySQL` está sendo adicionado ao MySQL 4.1 que também está disponível (versão alfa). Veja Seção 1.5.2 [MySQL 4.1 Nutshell], Página 24.

1.5.1.1 Recursos Disponíveis no MySQL 4.0

Aumento da velocidade

- O MySQL 4.0 tem uma cache de consultas que pode dar uma grande aumento na velocidade de aplicações com consutas repetitivas. Veja Seção 6.9 [Query Cache], Página 625.
- A versão 4.0 aumenta a velocidade do MySQL Server em um número e áreas tais como `INSERT`s em bloco, buscas em índices empacotados, criação de índices `FULLTEXT`, e `COUNT(DISTINCT)`.

Introdução ao Servidor MySQL Embutido

- A nova biblioteca do Servidor Ebutido pode ser facilmente usada em aplicações standalone e embarcadas. O servidor embutido fornce uma alternativa para o uso do MySQL em um ambiente cliente/servidor. Veja Seção 1.5.1.2 [Nutshell Embedded MySQL], Página 24.

Mecanismo de armazenamento InnoDB como padrão

- O mecanismo de armazenamento `InnoDB` é oferecido como um recurso padrão do servidor `MySQL`. Isto significa suporte a *transações* ACID, *chaves estrangeiras* com UPDATE/DELETE em cacata e *lock de registro* agora são recursos padrões. Veja Seção 7.5 [`InnoDB`], Página 643.

Novas fncionalidades

- A melhora das propriedades de busca `FULLTEXT` do MySQL Server 4.0 habilita indexação `FULLTEXT` de grandes partes de texto com linguagem natural e binária de lógica de busca. Você pode personalizar o tamanho mínimo de palavras e definir a sua própria lista de palavras de parasa em qualquer linguagem humana, habilitando um novo conjunto de aplicações a serem construídas no MySQL Server. Veja Seção 6.8 [Fulltext Search], Página 619.

Compatibilidade com os padrões, portabilidade e migração

- Recursos para simplificar a migração de outros sistemas de banco de dados para o MySQL Server incluem `TRUNCATE TABLE` (como no Oracle)
- Muitos usuários também ficarão satisfeitos ao perceber que o MySQL Server agora suporta a instrução `UNION`, um recurso padrão SQL muito esperado.
- O MySQL agora pode ser executado nativamente na plataforma Novell NetWare 6.0. Veja Seção 2.6.8 [Novell NetWare], Página 164.

Internacionalização

- Nossos usuários alemães, austríacos e suiços notarão que o `MySQL` agora suporta um novo conjunto de caracteres, `latin1_de`, que assegura que a `Ordenação em alemão` classificará palavras com umlauts na mesma ordem das agendas telefônicas alemãs.

Aprimoramento da Usabilidade

No porcesso de construção de recursos para novos usuários, não esquecemos os pedidos de nossa leal comunidade de usuários.

- A maioria dos parâmetros `mysqld` (opções de inicialização) agora podem ser definidas se finalizar o servidor. Isto é um recurso conveniente para Administradores de Bancos de Dados (DBAs). Veja Seção 5.5.6 [SET OPTION], Página 460.
- Instruções `DELETE` e `UPDATE` multi-tabelas foram adicionadas.
- Foi adicionado suporte ao mecanismo de armazenamento `MyISAM` para link simbólico no nível de tabela (e não apenas a nível de banco de dados como antes) e para habilitar o tratamento de links simbólicos no Windows por padrão.
- `SQL_CALC_FOUND_ROWS` e `FOUND_ROWS()` são novas funções que tornaram possível encontrar o números de linhas que uma consulta `SELECT` que inclui uma cláusula `LIMIT` teria retornado se a cláusula não fosse utilizada.

A seção de novidades deste manual inclui uma lista mais aprofundada dos recursos. Veja Seção D.3 [News-4.0.x], Página 966.

1.5.1.2 Servidor Embutido MySQL

`libmysqld` faz o MySQL Server adequado para uma grande área de aplicações. Usando a biblioteca do servidor MySQL embutido, pode embarcar o MySQL Server em vários aplicativos e dispositivos eletrônicos, onde o usuário final não têm conhecimento de possuir um banco de dados básico. O servidor MySQL embutido é ideal para uso nos bastidores em aplicações de Internet, quiosques públicos, responsável por unidades de combinação hardware/software, servidores Internet de alta performance, banco de dados de auto-contenção distribuídos em CDROM, e assim por diante

Muitos usuários da `libmysqld` se benficiarão da *iLicença Dual* do MySQL. Para aqueles que não desejam ser limitar pela `GPL`, o software é tambem está disponível sob uma licença comercial. A biblioteca embutida do MySQL também usa a mesma interface que a biblioteca do cliente normal, sendo então conveniente e fácil de usar. Veja Seção 12.1.15 [`libmysqld`], Página 864.

1.5.2 MySQL 4.1 in a Nutshell

MySQL Server 4.0 prepara a criação de novos recursos como subqueries e Unicode (implementado na versão 4.1) e o funcionamento de stored procedures do SQL-99 está sendo feito para a versão 5.0. Estes recursos estão no topo da lista de recursos desejados de muitos de nossos clientes.

Com estas adições, os críticos do MySQL Database Server devem ser mais imaginativos que nunca para apontar as deficiências do MySQL Database Management System. Já conhecido por sua estabilidadem velocidade e facilidade de uso, o MySQL Server estará apto a atender as necessidades de vários compradores exigentes.

1.5.2.1 Recursos Disponíveis no MySQL 4.1

Os recursos listados nesta seção estão implementados no MySQL 4.1. Outros poucos recursos estão planejados para o MySQL 4.1. Veja Seção 1.6.1 [TODO MySQL 4.1], Página 27.

A maioria dos novos recursos em codificação, como stored procedures, estarão disponíveis no MySQL 5.0. Veja Seção 1.6.2 [TODO MySQL 5.0], Página 27.

Suporte a subqueries e tabelas derivadas

- Uma subquery é uma instrução `SELECT` aninhada dentro de outras instruções. Uma tabela dericada (unnamed view) é uma subquery na cláusula `FROM` de outras instruções. Veja Seção 6.4.2 [Subqueries], Página 570.

Aumento na velocidade

- Protocols binários mais rápidos com instruções preparadas e parâmetros de ligação. Veja Seção 12.1.4 [C API Prepared statements], Página 828.
- Indexação `BTREE` agora é suportado por tabelas `HEAP`, aumentando de forma significante o tempo de resposta para busca que não são exatas.

Nova funcionalidade

- `CREATE TABLE tabela1 LIKE tabela2` lhe permite criar uma nova tabela com a estrutura exatamente igual a de uma tabela existente, usando um único comando.
- Suporte aos tipos espaciais OpenGIS (dados geográficos). Veja Cap“ptexi tulo 10 [Spatial extensions in MySQL], Página 733.
- A replicação pode ser feita sobre conexão SSL.

Compatibilidade aos padrões, portabilidade e migração

- O novo protocolo cliente/servidor adiciona a possibilidade de se passar múltiplos avisos ao cliente, no lugar se um único resultado. Isto faz com que o trabalho como uma grande carga de dados seja muito mais fácil de rastrear. `SHOW WARNINGS` exibe avisos para o último comando. Veja Seção 4.6.8.9 [`SHOW WARNINGS`], Página 323.

Internacionalização

- Para suportar nossa base de usuário sempre em expansão usando linguagens locais nas aplicações, o programa MySQL agora oferece suporte Unicode extensivo por meio dos conjunto de caracteres `utf8` e `ucs2`.
- Os conjuntos de caracteres agora podem ser definidos por colunas, tabelas e banco de dados. Isto permite um alto grau de flexibilidade no desenho das aplicações, particularmente para sites-web multi-linguagens.
- Para documentação sobre este suporte a conjunto de caracters aprimorados, veja Cap“ptexi tulo 9 [Charset], Página 710.

Aprimoramento da usabilidade

- Em resposta a demanda popular, adicionamos um comando `HELP` com base no servidor que pode ser usado para conseguir ajuda para comandos MySQL. A vantagem de se ter esta informação no lado do servidor é que a informação é sempre aplicável para aquela versão do servidor em particular. Como esta informação está disponível executando uma instrução SQL, outros clientes também poderão ser escritos para acessá-la. Por exemplo, o cliente `mysql` de linha de comando foi modificado para ter esta capacidade.

- No novo protocolo cliente/servidor, várias instruções podem ser feitas com uma única chamada. Veja Seção 12.1.8 [C API multiple queries], Página 855.
- O novo protocolo cliente/servidor também suporta retorno de vários resultados. Isto pode ocorrer como resultado de enviar várias instruções, por exemplo (Veja o item anterior).
- Uma nova sintaxe `INSERT ... ON DUPLICATE KEY UPDATE ...` tem sido implementada. Isto lhe permite executar um `UPDATE` em um registro existente se o um `INSERT` criasse uma chave (índice) primária (`PRIMARY`) ou única (`UNIQUE`) (index) duplicada. Veja Seção 6.4.3 [INSERT], Página 579.
- Projetamos uma nova função de agrupamento `GROUP_CONCAT()`, adicionando a capacidade de concatenar colunas de registros agrupados em uma única string de resultado, o que é extremamente útil. Veja Seção 6.3.7 [Group by functions and modifiers], Página 555.

A seção de novidades neste manual incluem uma lista mais completa de recursos. Veja Seção D.2 [Novidades na versão 4.1.x], Página 957.

1.5.2.2 Stepwise Rollout

Novos recursos estão sendo adicionados ao MySQL 4.1. A versão Alfa já stá disponível para download. Veja Seção 1.5.2.3 [Nutshell Ready for Immediate Use], Página 26.

O conjunto de recursos que estão sendo adicionados a versão 4.1 estão, na maioria, corrigidos. Desenvolvimento adicional já está em andamento na versão 5.0. O MySQL 4.1 passam pelos passos de *Alfa* (tempo no qual os novos recursos ainda podem ser adionados/alterados), *Beta* (quando já implementamos todos os recursos e apenas correções de erros são realizados0) e *Gamma* (indicando que ima distribuição de produção está quase pronta). No fim deste processo, o MySQL 4.1 se tornará o nova distribuição de produção).

1.5.2.3 Pronto para Uso em Desenvolvimento Imediato

O MySQL 4.1 está atualmente no estágio alfa e os binários estão disponíveis para download em `http://www.mysql.com/downloads/mysql-4.1.html`. Todas as distribuições binárias passaram por nossos extensivos teste sem nenhum erro na plataforma em que testamos. Veja Seção D.2 [Novidades na versão 4.1], Página 957.

Para aqueles que desejam usar o fonte mais recente do desenvolvimento do MySQL 4.1, deixamos nosso repositório do BitKeeper publicamente disponível. Veja Seção 2.3.4 [Installing source tree], Página 100.

1.5.3 MySQL 5.0, A Próxima Distribuição de Desenvolvimento

O novo desenvolvimento para o MySQL está focado na distribuição 5.0, comrecursos como Stored Procedures entre outros. Veja Seção 1.6.2 [TODO MySQL 5.0], Página 27.

Para aqueles que desejam dar uma olhada no desenvolvimento do MySQL, deixamos o nosso repositórioo do BitKeeper para o MySQL versão 5.0 disponível publicamente. Veja Seção 2.3.4 [Instalando a árvore fonte], Página 100.

1.6 MySQL e o Futuro (o TODO)

Esta seção lista os recursos que planejamos impementar no `MySQL Server`. As listas são apresentadas por versão, e os itens estão aproximadamente na ordem em que serão feitos.

Nota: Se você é um usuário corporativo com uma necessidade urgente de um recurso particular, por favor, contate `sales@mysql.com` para conversarmos sobre patrocínio. Financiamento feito por uma ou mais companhias nos permite alocar recursos adicionais para aquele propósito específico. Um exemplo de um recurso patrocinado no passado é a replicação.

1.6.1 Novos Recursos Planejados Para a Versão 4.1

Os recursos abaixo ainda não estão implementados no MySQL 4.1, mass estão planejados para implementação antes que o MySQL 4.1 vá para a fase beta. Para uma lista do que já está feito no MySQL 4.1, veja Seção 1.5.2.1 [Nutshell 4.1 features], Página 24.

- Suporte OpenSSL estável (o MySQL 4.0 tem suporte rudimentar ao OpenSSL, não testado 100%).
- Mais teste de instruções preparadas
- Mais testes de múltiplos conjunto de caracteres para uma tabela.

1.6.2 Novos Recursos Planejados Para a Versão 5.0

Os seguintes recursos estão planejados para inclusão no MySQL 5.0. Note que como possuimos diversos desenvolvedores que estão trabalhando em diferentes projetos, haverão também muitos recursos adicionais. Há também um pequena chance qie alguns destes recursos sejam adicionados ao MySQL 4.1. Para uma lista do que já está feito no MySQL 4.1, veja Seção 1.5.2.1 [Nutshell 4.1 features], Página 24.

Para aqueles que desejam dar uma olhada nas novidades do desenvolvimento do MySQL, deixamos nosso repositório BitKeeper para o MySQL versão 5.0 publicamente disponível. Veja Seção 2.3.4 [Instalando a árvore fonte], Página 100.

Stored Procedures

- Stored procedures estão sendo implementadas atualmente. Este esforço é baseado no SQL-99, o que tem m sintaxe básica similar (mas não idêntica) a do Oracle PL/SQL. Nós também implementaremos o framework do SQL-99 para enganchar em linguagens externas e (onde possível) compatibilidade com p.ex. PL/SQL e T-SQL.

Nova funcionalidade

- Suporte a cursores elementares.
- A habilidade de especificar explicitamente para tabelas `MyISAM` que um índice deve ser criado como um índice `RTREE`. Na versão 4.1, índices `RTREE` são usados internamente para dados geométricos (tipos de dados GIS), mas não podem ser criados no pedido.
- Registros de tamanhos dinâmicas para tabelas `HEAP`.

Compatibilidade com o padrão, portabilidade e migração

- Adiciona suporte real a `VARCHAR` (tamanho de colunas maiores que 255, e sem corte de espaços em branco extras). (Já existe suporte para isto nos mecanismos de armazenamento do `MyISAM`, mas ainda não está disponível a nível de usuário).

Aumento na velocidade

- `SHOW COLUMNS FROM nome_tabela` (usado pelo cliente `mysql` para permitir expansões de nomes de colunas) não deve abrir a tabela, apenas o arquivo de definição. ISto exigirá menos memória e será muito mais rápido.
- Permite que o `DELETE` em tabelas `MyISAM` usem a cache de registros. Para fazer isto, precisamos atualizar a thread da cache de registro quando atualizarmos os arquivos '`.MYD`'.
- Melhores tabes em memória (`HEAP`):
  - Registro de tamanhos dinâmoicos.
  - Tratamento de registro mais rápido (menos cópia).

Internacionalização

- Ap usar `SET CHARACTER SET` devemos traduzir toda a consulta de uma vez e não apenas as strings. Isto permitirá que os usuários usem caracteres traduzidos nos nomes de banco de dados, tabelas e colunas.

Aprimoramento da usabilidade

- Resolver a questão de `RENAME TABLE` em uma tabela usada em uma tabela `MERGE` ativa, o que possivelmente corrompe a tabela.

1.6.3 Novos Recursos Planejados Para a Versão 5.1

Novas funcionalidades

- Suporte `FOREIGN KEY` para todos os tipos de tabelas.
- Restrições a nível de colunas.
- Replicação seguro a falhas.
- Backup online com baixa queda de desempenho. O backup online tornará mais fácil adicionar um novo slave de replicação sem desligar o master.

Aumento de velocidade

- Novo formato dos arquivos de definição e tabelas baseados em texto (arquivos '`.frm`') e uma cache de tabelas para a definição de tabelas. Isto nos permitirá fazer consultas mais rápidas da estruturas de tabela e dar um suporte a chaves estrangeiras mais eficiente.
- Otimizar o tipo `BIT` para gastar 1 bit (agora `BIT` gasta 1 byte; e é tratado como um sinônimo para `TINYINT`.)

Aprimoramento da usabilidade

- Adicionar opções ao protocolo cliente/servidor para obter notas de progresso para longos comandos em execução.
- Implementar `RENAME DATABASE`. Para tornar isto seguro para todos os mecanismos de armazenamento, ele deve funcionar como a seguir:

    - Cria um novo banco de dados.
    - Para cada tabelas, renomeie-a para outro banco de dados, o qual fazemos com o comando `RENAME`.
    - Apagar o banco de dados antigo.
- Nova alteração da interface de arquivo interno. Isto fará todos os manipuladores de arquivos mais gerais e tornará mais fácil adicionar extensões tipo RAID.

1.6.4 Novos Recursos Planejados Para a Versão em um Futuro Próximo

Novas funcionalidade

- Comando como do Oracle `CONNECT BY PRIOR ...` para estruturas de busca tipo árvore (hierárquica)
- Adicionar todos os tipos que faltam do SQL-92 e ODBC 3.0.
- Adicionar `SUM(DISTINCT)`.
- `INSERT SQL_CONCURRENT` e `mysqld --concurrent-insert` para fazer uma inserção concorrente no fim do arquivo se o arquivo tiver lock de leitura.
- Permitir a atualização de variáveis nas instruções `UPDATE`. Por exemplo: `UPDATE TABLE foo SET @a=a+b,a=@a, b=@a+c`.
- Alterar quando as variáveis de usuários são atualizadas e assim pode se usá-las com `GROUP BY`, como no exemplo a seguir: `SELECT id, @a:=COUNT(*), SUM(sum_col)/@a FROM nome_tabela GROUP BY id`.
- Adicionar a opção `IMAGE` a `LOAD DATA INFILE` para não atualizar campos `TIMESTAMP` e `AUTO_INCREMENT`.
- Adicionar a sintaxe `LOAD DATA INFILE ... UPDATE` que funciona assim:
    - Para tabelas com chaves primárias, se o registro de entrada contém um valor de chave primária, linhas existentes correspondendo às chaves primárias são atualizadas para o restante das colunas de entrada. No entanto, colunas `faltosas` na inserção dos registros de entradas não são alteradas.
    - Para tabelas com chaves primárias, se um registro de entrada não contém um valor de chave primária ou estrá faltando alguma parte da chave, o registro é tratado como um `LOAD DATA INFILE ... REPLACE INTO`.
- Fazer com que `LOAD DATA INFILE` entenda a sintaxe do tipo:

```
LOAD DATA INFILE 'file_name.txt' INTO TABLE tbl_name
     TEXT_FIELDS (text_field1, text_field2, text_field3)
     SET table_field1=CONCAT(text_field1, text_field2),
         table_field3=23
     IGNORE text_field3
```

  Isto pode ser usado para saltar colunas extras no arquivo texto, ou atualizar colunas baseadas nas expressões dos dados lidos.

- Novas funções para tyrabalhar com tipos de colunas `SET`:
  - `ADD_TO_SET(valor,conjunto)`
  - `REMOVE_FROM_SET(valor,conjunto)`
- Se você abortar o `mysql` no meio de uma consulta, você deve abrir outra conexão e matar a consulta antiga em execução. Alternativamente, deve ser feita um tentativa de detecção deste problema no servidor.
- Adicione um interface do mecanismo de armazenamento para informações da tabela assim que você puder usá-la como uma tabela de sistema. Isto seria um pouco mais lento se você pedisse informações sobre todas as tabelas, mas muito flexível. `SHOW INFO FROM tbl_name` para informações básicas das tabelas deve ser implementado.
- Permite `SELECT a FROM crash_me LEFT JOIN crash_me2 USING (a)`; neste caso é considerado que `a` vem da tabela `crash_me`.
- Opções `DELETE` e `REPLACE` para a instrução `UPDATE` (isto deletará registros quando se tiver um erro de chave duplicada durante a atualização).
- Altera o formato de `DATETIME` para armazenar frações de segundo.
- Possibilitar o uso da nova biblioteca regexp GNU em vez da atual (a biblioteca GNU deve ser muito mais rápida que a antiga).

Compatibilidade com os padrões, portabilidade e migração

- Não adicionar valores `DEFAULT` automáticos as colunas. Enviar um erro ao usar um `INSERT` que não contenha uma coluna que não tenha um `DEFAULT`.
- Adicionar as funções de agrupamento `ANY()`, `EVERY()` e `SOME()`. No padrão SQL isto só funciona em colunas booleanas, mas podemos extendê-las para funcionar em qualquer coluna/expressão tratando valores 0 como FALSE e valores diferentes de 0 como TRUE.
- Corrigir para que o tipo de `MAX(coluna)` seja o mesmo do tipo da coluna:

```
mysql> CREATE TABLE t1 (a DATE);
mysql> INSERT INTO t1 VALUES (NOW());
mysql> CREATE TABLE t2 SELECT MAX(a) FROM t1;
mysql> SHOW COLUMNS FROM t2;
```

Aumento de velocidade

- Não permitir mais que um número definido de threads façam a recuperação do MyISAM ao mesmo tempo.
- Alterar `INSERT ... SELECT` para usar inserções concorrentes opcionalmente.
- Adicionar uma opção para descarregar paginas de chaves para tabelas com delayed keys se elas não forem usados por um tempo.
- Permitir joins em partes de chaves (otimização).
- Adicionar simulação de `pread()`/`pwrite()` no Windows para permitir inserções concorrentes.
- Um analizador de arquivos de log que possam analizar informações sobre quais tabelas são usadas com mais frequência, a frequência com que joins

multi-tables são executados, etc. Isto deve ajudar os usuários a identificar áreas ou projetos de tabelas que podiam ser otimizados para executar consultas muito mais eficientes.

Internacionalização

Aprimoramentos de usabilidade

- Retorna os tipos dos campos originais ao se fazer `SELECT MIN(coluna) ... GROUP BY`.
- Possibilita especificar `long_query_time` com uma granularidade em microsegundos.
- Ligue o código `myisampack` no servidor assim ele poderá realizar operações `PACK` e `COMPRESS`.
- Adicionar uma cache de chaves temporária durante `INSERT/DELETE/UPDATE` para podermos fazer um recuperação se o índice ficar cheio.
- Se você realizar um `ALTER TABLE` em uma tabela que é ligada simbolicamente a outro disco, crie tabelas tenporárias neste disco.
- Implementar um tipo `DATE/DATETIME` que trate as informações de fusos horários de forma apropriada e assim lidar com datas em diferentes fusos horários será mais fácil.
- Corrigir o configure para se poder compilar todas as bibliotecas (como no `MyISAM`) sem threads.
- Permitir variáveis SQL em `LIMIT`, como em `LIMIT @a,@b`.
- Saída automática do `mysql` para um navegador web.
- `LOCK DATABASES` (com diversas opções).
- Muito mais variáveis para `SHOW STATUS`. Leitura e atualização de registros. Selects em 1 tabela e select com joins. Número de tabelas na select. Número de consultas `ORDER BY` e `GROUP BY`.
- `mysqladmin copy database novo-banco_dados`; exige que o comando `COPY` seja adicionado ao `mysqld`.
- Lista de processos deve mostar o número de consultas/threads.
- `SHOW HOSTS` para xibir informações sobre a cache de nome de máquina.
- Alterar o nome de tabelas de string vazias para `NULL` para colunas calculadas.
- Não usar `Item_copy_string` em valores numéricos para evitar a conversão number->string->number no casos de: `SELECT COUNT(*)*(id+0) FROM nome_tabela GROUP BY id`
- Alterar aqueles `ALTER TABLE` que não abortam clientes que executam `INSERT DELAYED`.
- Colunas referênciadas em uma cláusula `UPDATE` irão conter os valores antigos antes da atualização iniciar.

Novos sistemas operacioais.

- Portar os clientes MySQL para LynxOS.

1.6.5 Novos Recursos Planejados Para a Versão em um Futuro a Médio Prazo

- Implementar função: `get_changed_tables(timeout,table1,table2,...)`
- Alterar leitura através de tabelas para usar mapeamento de memória quando possível. Atualmente somente tabelas compactadas usam mapeamento de memória.
- Tornar o código de timestamp automático melhor. Adicionar timestamps para o log de atualizações com `SET TIMESTAMP=#;`
- Usar mutex de leitura/escrita em alguns lugares para obter maior velocidade.
- Views simples (inicialmente em uma tabela, depois em qualquer expressão). Veja Seção 1.8.4.6 [ANSI diff Views], Página 51.
- Fechar algumas tabelas automaticamente se uma tabela, tabela temporária ou arquivos temporários obtiverem o erro 23 (não pode abrir arquivos suficientes).
- Melhor propagação de constantes. Quando uma ocorrência de `nome_col=n` é encontrada em uma expressão, para algumas constantes `n`, substitua outras ocorrências de `nome_col` dentro da expressão por `n`. Atualmente, isto é feito somente para alguns casos simples.
- Alterar todas expressões const com expressões calculadas se possível.
- Chave otimizadora = expressão. No momento somente a chave = campo ou a chave = constante são otimizadas.
- Melhorar o código de algumas das funções de cópia
- Alterar '`sql_yacc.yy`' para um analizador em linha para reduzir seu tamanho e obter melhores mensagems de erro (5 dias).
- Alterar o analisador para usar somente uma regra para diferentes números de argumentos em uma função.
- Utilizar nomes de cálculo completos na parte de ordenação. (For ACCESS97)
- `MINUS`, `INTERSECT` e `FULL OUTER JOIN`. (Atualmente `UNION` [na 4.0] e `LEFT OUTER JOIN` são suportados).
- `SQL_OPTION MAX_SELECT_TIME=#` para colocar um limite de tempo em uma pesquisa.
- Fazer o log de atualizações gravar em um banco de dados.
- `LIMIT` negativo para recuperar dados do fim.
- Alarmes em funções clientes de conexão, leitura e escrita.
- Por favor, perceba as alterações ao `mysqld_safe`: de acordo com o FSSTND (que o Debian tenta seguir) arquivos PID dever ir em '`/var/run/<progname>.pid`' e arquivos de log em '`/var/log`'. Seria ótimo se você puder colocar o diretório de dados na primeira declaração de "pidfile" e "log", para que a colocação destes arquivos possa ser alterada com uma simples instrução.
- Permitir um cliente requisitar log.
- Adicionar uso de `zlib()` a `LOAD DATA INFILE`, para permitir que as instruções leiam arquivos compactados com `gzip`.
- Corrigir ordenação e agrupamento de colunas `BLOB` (parcialmente resolvida agora).
- Alterar para o uso de semáforos quando contar threads. Devemos primeiro implementar uma biblioteca de semáforos para a MIT-pthreads.

- Adicionar suporte pleno para `JOIN` com parênteses.
- Como uma alternativa para uma thread / conexão gerencie uma fila de threads para manipular as pesquisas.
- Permitir obter mais de um bloqueio com `GET_LOCK`. Quando isto for feito, serão, também, tratados os possíveis deadlocks que essa alteração irá acarretar.

O tempo é fornecido de acordo com a quantidade de trabalho, e não tempo real.

1.6.6 Novos Recursos que Não Planejamos Fazer

- Nada; Planejamos ser totalmente compatíveis com o ANSI 92 / ANSI 99.

1.7 Fontes de Informações do MySQL

1.7.1 Listas de Discussão MySQL

Esta seção introduz a lista de deiscussão do MySQL e dá algumas explicações sobre como a lista deve ser utilizada. Quando você se inscreve na lista de discussão, você receberá, como mensagens de email, tudo o que é enviado para a lista. Você também poderá enviar suas próprias dúvidas e respostas para a lista.

1.7.1.1 As Listas de Discussão do MySQL

Para se inscrever ou cancelar a inscrição de qualquer uma das listas de email descritas nesta seção, visite `http://lists.mysql.com/`. Por favor, **não** envie mensagem sobre inscrição ou cancelamento para qualquer das listas de emasil, porque tais mensagens são distribuídas automaticamente para milhares de outros usuários.

Seu site local pode ter muitas inscrições para uma lista de email do MySQL. Se sim, o site pode ter uma lista de email local, assim as mensagens enviadas para `lists.mysql.com` do seu site são propagadas para a lista local. Nestes casos, por favor, contate seu administrador de sistema para adicionado ou excluido da lista local do MySQL.

Se você quiser que as mensagens da lista de discussão sejam enceminhadas para uma caixa de correio separada no seu programa de emails, configure um filtro com base nos cabeçalhos das mensagens. Você pode também usar os cabeçalhos `List-ID:` ou `Entregar-Para:` para identificar suas mensagens.

Existe também as seguintes listas de discussão sobre MySQL atualmente:

`announce` Esta é para anuncio de novas versões do MySQL e programas relacionados. Esta é uma lista com baixo volume na qual todos usuarios do MySQL deveriam se inscrever.

`mysql` A principal lista para discussões MySQL em geral. Note que alguns tópicos são mais bem discutidos em listas mais especializadas. Se você enviar para a lista errada você pode não obter resposta.

`mysql-digest`
: A lista mysql na forma resumida. Isto significa que você irá receber todas mensagens individuais, enviadas na forma de uma grande mensagem uma única vez ao dia.

`bugs`
: Esta lista só será do seu interesse se você quiser ficar informado sobre assuntos relatados desde a última distribuição do `MySQL` ou se você quiser estar ativamente envolvido no processo de busca e correção de erros. Veja Seção 1.7.1.3 [Relatório de erros], Página 36.

`bugs-digest`
: Uma versão resumida da lista `bugs`.

`internals`
: Uma lista para pessoas que trabalham no código do MySQL. Nesta lista pode-se discutir desenvolvimento do MySQL e pos-patches.

`internals`
: Uma versão resumida da lista `internals`.

`mysqldoc`
: Esta lista é para pessoas que trabalham na documentação do MySQL: pessoas da MySQL AB, tradutores e outros membros da comunidade.

`mysqldoc-digest`
: Esta é uma versão resumida da lista `mysqldoc`.

`benchmarks`
: Esta lista é para qualquer um interessado em assuntos de desempenho. Discussões concentradas em desempenho de banco de dados (não limitado ao MySQL) mas também inclue categorias ,com desempenho do kernel, sistema de arquivos, sistema de disco e outros.

`benchmarks`
: Esta é uma versão resumida da lista `benchmarks`.

`packagers`
: Esta lista é para discussões sobre empacotamento e distribuição do MySQL. Este é o fórum usado pela pessoas que mantém a distribuição para troca de idéias de pacotes do MySQL e para assegurar que o MySQL esteja o mais parecido possível em todas as plataformas e sistemas operacionais suportados.

`packagers-digest`
: Esta é uma versão resumida da lista `packagers`.

`java`
: Discussão sobre o servidor MySQL e Java. É mais usada para discussões sobre o driver JDBC, incluindo MySQL Connector/J.

`java-digest`
: Uma versão resumida da lista `java`.

`win32`
: Esta é a lista para todos os tópicos relacionados ao MySQL em sistemas operacionais Microsoft, como o Win95, Win98, NT e Win2000.

`win32-digest`
: Uma versão resumida da lista `win32`.

`myodbc` Lista para todos os tópicos relacionados a conectividade do MySQL com ODBC.

`myodbc-digest`
Uma versão resumida da lista `myodbc`.

`mysqlcc` Esta lista é sobre todos os tópicos relativos ao cliente gráfico `MySQL Control Center`.

`mysqlcc-digest`
Esta lista é uma versão resumida da lista `mysqlcc`.

`plusplus` Lista sobre todos os tópicos relacionados à programação da API C++ para o MySQL.

`plusplus-digest`
Uma versão resumida da lista `plusplus`.

`msql-mysql-modules`
Lista sobre o Suporte MySQL no Perl com o `msql-mysql-modules` que é chamado `DBD-mysql`.

`msql-mysql-modules-digest`
Lista resumida sobre a versão do `msql-mysql-modules`.

Se você não obtiver uma resposta para suas questões na lista de mensagens do `MySQL`, uma opção é pagar pelo suporte da MySQL AB, que irá colocar você em contato direto com desenvolvedores MySQL. Veja Seção 1.4.1 [Suporte], Página 17.

A seguinte tabela mostra algumas listas de mensagens sobre o MySQL que utilizam linguas diferentes do Inglês. Perceba que elas não são operadas pela MySQL AB, portanto, não podemos garantir a qualidade destas.

`mysql-france-subscribe@yahoogroups.com Lista de mensagens na`
língua francesa.

`list@tinc.net Lista de mensagens coreana.`
Envie `subscribe mysql your@email.address` para esta lista.

`mysql-de-request@lists.4t2.com Lista de mensagens alemã.`
Envie `subscribe mysql-de your@email.address` para esta lista. Você pode encontrar informações sobre esta lista de mensagens em `http://www.4t2.com/mysql`.

`mysql-br-request@listas.linkway.com.br Lista de mensagens`
em português Envie `subscribe mysql-br your@email.address` para esta lista.

`mysql-alta@elistas.net Lista de mensagens espanhola.`
Envie `subscribe mysql your@email.address` para esta lista.

1.7.1.2 Fazendo perguntas ou relatando erros

Antes de enviar um relato de erro ou uma questão, por favor faça o seguinte:

- Comece pesquisando o manual MySQL online em:
`http://www.mysql.com/doc/`

Nós tentaremos manter o manual atualizado, frequentemente atualizando-o com soluções para novos problemas encontrados! O apêndice de histórico de mudanças (`http://www.mysql.com/doc/en/News.html`) pode ser útil já que é bem possível que uma versão mais nova ja tenha a solução para o seu problema.

- Procure no banco de dados de bugs em `http://bugs.mysql.com/` para ver se o erro já foi relatado/resolvido.
- Pesquise os arquivos das listas de mensagens MySQL: `http://lists.mysql.com/`

- Você pode também usar a página `http://www.mysql.com/search.html` para pesquisar todas as páginas Web (incluindo o manual) que estão localizados em `http://www.mysql.com/`.

Se você não puder encontrar uma resposta no manual ou nos arquivos, confira com seu expert em MySQL local. Se você continua não encontrando uma resposta para sua questão, vá em frente e leia a próxima seção para saber como enviar email para lista de email do MySQL.

1.7.1.3 Como relatar erros ou problemas

Nosso banco de dados de bugs é publico e pode ser pesquisado por qualquer um em `http://bugs.mysql.com/`. Se você logar no sistema, você poderá entrar novos relatórios.

Escrever um bom relatório de erro exige paciência, e fazê-lo de forma apropriada economiza tempo para nós e para você. Um bom relatório de erros contendo um teste de caso para o bug irá torná-lo muito mais fácil para corrigí-lo no próximo release. Esta seção irá ajudá-lo a escrever seu relatório corretamente para que você não perca seu tempo fazendo coisas que não irão ajudar-nos muito ou nada.

Nós encorajamos todo mundo a usar o script `mysqlbug` para gerar um relato de erros (ou um relato sobre qualquer problema), se possível. `mysqlbug` pode ser encontrado no diretório '`scripts`' na distribuição fonte, ou, para uma distribuição binária, no diretório '`bin`' no diretório de instalação do MySQL. Se você não puder utilizar o `mysqlbug` (por exemplo, se você o estiver executando no Windows), é ainda de vital importância que você incluia todas as informações necessárias listadas nesta seção (o mais importante é uma descrição do sistema operacional e a versão do MySQL).

O script `mysqlbug` lhe ajudará a gerar um relatório determinando muitas das seguintes informações automaticamente, mas se alguma coisa importante estiver faltando, por favor forneça-o junto de sua mensagem! Por favor leita esta seção com cuidado e tenha certeza que todas as informações descritas aquie estão incluídas no seu relatório.

De preferência, você deve testar o problema usando a última versão de produção ou desenvolvimento do Servidro MySQL antes do envio. Qualquer um deve estar apto a repetir o erro apenas usando '`mysql test < script`' no caso de teste incluido ou executando o script sheel ou Perl que é incluído no relatório de erros.

Todos os erros enviados para o banco de dados dem bugs em `http://bugs.mysql.com/` serão corrigidos ou documentados na próxma distribuição do MySQL. Se apenas pequenas mudanças de código forem necessárias enviaremos um patch para corrigir o problema.

O lugar comum para relatar erros e problemas é `http://bugs.mysql.com`.

Se você encontrar um erro de segurança no MySQL, envie um email para `security@mysql.com`.

Se você tiver um relatório de erro que possa ser repetido, relate-o no banco de dados de bugs em `http://bugs.mysql.com`. Note que mesmo neste caso é bom executar o script `mysqlbug` primeiro para ter informações sobre o sistema. Qualquer erro que pudermos repetir tem uma grande chance de ser corrigido na próxima distribuição do MySQL.

Para relatar outros problemas, você pode usar a lista de email do MySQL.

Lembre-se que é possível responder a uma mensagem contendo muita informação, mas não a uma contendo muito pouca. Frequentemente pessoas omitem fatos porque acreditam que conhecem a causa do problema e assumem que alguns detalhes não importam. Um bom principio é: Se você está em dúvida sobre declarar alguma coisa, declare-a ! É milhares de vezes mais rápido e menos problemático escrever um pouco de linhas a mais no seu relatório do que ser forçado a perguntar de novo e esperar pela resposta porque você não forneceu informação sufiente da primeira vez.

Os erros mais comuns acontecem porque as pessoas não indicam o número da versão da distribuição do MySQL que estão usando, ou não indicam em qual plataforma elas tem o MySQL instalado (Incluindo o número da versão da plataforma). Essa informação é muito relevante, e em 99% dos casos o relato de erro é inútil sem ela! Frequentemente nós recebemos questões como, "Por que isto não funciona para mim?" então nós vemos que aquele recurso requisitado não estava implementado naquela versão do MySQL, ou que o erro descrito num relatório foi resolvido em uma versão do MySQL mais nova. Algumas vezes o erro é dependente da plataforma; nesses casos, é quase impossível corrigir alguma coisa sem conhecimento do sistema operacional e o número da versão da plataforma.

Lembre-se também de fornecer informações sobre seu compilador, se isto for relacionado ao problema. Frequentemente pessoas encontram erros em compiladores e acreditam que o problema é relacionado ao MySQL. A maioria dos compiladores estão sobre desenvolvimento todo o tempo e tornam-se melhores a cada versão. Para determinar se o seu problema depende ou não do compilador, nós precisamos saber qual compilador foi usado. Note que todo problema de compilação deve ser estimado como relato de erros e, consequentemente publicado.

É de grande ajuda quando uma boa descrição do problema é incluída no relato do erro. Isto é, um bom exemplo de todas as coisas que o levou ao problema e a correta descrição do problema. Os melhores relatórios são aqueles que incluem um exemplo completo mostrando como reproduzir o erro ou o problema Veja Seção E.1.6 [Casos de teste reproduzíveis], Página 1084.

Se um programa produz uma mensagem de erro, é muito importante incluir essas mensagens no seu relatório! Se nós tentarmos procurar por algo dos arquivos usando programas, é melhor que as mensagens de erro relatadas sejam exatamente iguais a que o programa produziu. (Até o caso deve ser observado!) Você nunca deve tentar lembrar qual foi a mensagem de erro; e sim, copiar e colar a mensagem inteira no seu relatório!

Se você tem um problema com o MyODBC, você deve tentar gerar um arquivo para rastremento de erros (trace) do MyODBC. Veja Seção 12.2.7 [MyODBC bug report], Página 879.

Por favor lembre-se que muitas das pessoas que lerão seu relatório podem usar um vídeo de 80 colunas. Quando estiver gerando relatórios ou exemplos usando a ferramenta de linha de comando `mysql`, então deverá usar a opção `--vertical` (ou a instrução terminadora `\G`) para saída que irá exceder a largura disponível para este tipo de vídeo (por exemplo, com a instrução `EXPLAIN SELECT`; veja exemplo abaixo).

Por favor inclua a seguinte informação no seu relatório:

- O número da versão da distribuição do MySQL que está em uso (por exemplo, MySQL Version 3.22.22). Você poderá saber qual versão vocês está executando, usando o comando `mysqladmin version`. `mysqladmin` pode ser encontrado no diretório '`bin`' sob sua instalação do MySQL.
- O fabricante e o modelo da máquina na qual você está trabalhando.
- O nome do sistema operacional e a versão. Para a maioria dos sistemas operacionais, você pode obter esta informação executando o comando Unix `uname -a`. Se você trabalho no Windows, você pode normalmente conseguir o nome e o número da versão com um duplo clique sobre o ícone "Meu Computador" e em seguida no menu "Ajuda/Sobre o Windows".
- Algumas vezes a quantidade de memória (real e virtual) é relevante. Se estiver em dúvida, inclua esses valores.
- Se você estiver usando uma distribuição fonte do MySQL, é necessário o nome e número da versão do compilador usado. Se você estiver usando uma distribuição binária, é necessário o nome da distribuição.
- Se o problema ocorre durante a compilação, inclua a(s) exata(s) mensagem(s) de erro(s) e também algumas linhas do contexto envolvendo o código no arquivo onde o erro ocorreu.
- Se o `mysqld` finalizou, você deverá relatar também a consulta que travou o `mysqld`. Normalmente você pode encontrar isto executando `mysqld` com o log habilitado. Veja Seção E.1.5 [Using log files], Página 1083.
- Se alguma tabela do banco de dados estiver relacionado ao problema, inclua a saída de `mysqldump --nodata nome_db nome_tbl1 nome_tbl2...`. Isto é muito fácil de fazer e é um modo poderoso de obter informações sobre qualquer tabela em um banco de dados que irá ajudar-nos a criar uma situação parecida da que você tem.
- Para problemas relacionados à velocidade ou problemas com instruções `SELECT`, você sempre deve incluir a saída de `EXPLAIN SELECT ...` e ao menos o número de linhas que a instrução `SELECT` produz. Você também deve incluir a saída de `SHOW CREATE TABLE nome_tabela` para cada tabela envolvida. Quanto mais informação você fornecer sobre a sua situação, mais fácil será para alguém ajudar-lo! A seguir um exemplo de um relatório de erros muito bom (ele deve ser postado com o script `mysqlbug`):

  Exemplo de execução usando a ferramenta de linha de comando `mysql` (perceba o uso do instrução terminadora `\G` para instruções cuja largura de saída deva ultrapassar 80 colunas):

```
mysql> SHOW VARIABLES;
mysql> SHOW COLUMNS FROM ...\G
       <saida para SHOW COLUMNS>
mysql> EXPLAIN SELECT ...\G
```

```
        <saida para EXPLAIN>
mysql> FLUSH STATUS;
mysql> SELECT ...;
        <Uma pequena versão da saída do SELECT,
        incluindo a hora em que a consulta foi executada>
mysql> SHOW STATUS;
        <saida do SHOW STATUS>
```

- Se um erro ou problema ocorrer quando estiver executando o **mysqld**, tente fornecer um script de entrada que irá reproduzir a anomalia. Este script deve incluir qualquer arquivo de fonte necessário. Quanto mais próximo o script puder reproduzir sua situação, melhor. Se você puder fazer uma série de testes repetidos, você poderá postá-lo para o `bugs@lists.mysql.com` para um tratamento de alta prioridade!

  Se não puder fornecer o script, você ao menos deve incluir a saída de `mysqladmin variables extended-status processlist` na sua mensagem para fornecer alguma informação da performance do seus sistema.

-

  Se você não puder produzir um caso de teste em algumas linhas, ou se a tabela de testes for muito grande para ser enviada por email para a lista de mensagens (mais de 10 linhas), você deverá dar um dump de suas tabelas usando o `mysqldump` e criar um arquivo '`README`' que descreve seu problema.

  Crie um arquivo comprimido de seus arquivos usando `tar` e `gzip` ou `zip`, e use o `ftp` para transferir o arquivo para `ftp://support.mysql.com/pub/mysql/secret/`. E envie uma pequena descrição do problema para `bugs@lists.mysql.com`.

- Se você achar que o MySQL produziu um resultado estranho para uma consulta, não inclua somente o resultado, mas também sua opinião de como o resultado deve ser, e uma conta descrevendo o base de sua opinião.
- Quando fornecer um exemplo do problema, é melhor usar os nomes de variáveis, nomes de tabelas, etc. utilizados na sua situação atual do que enviar com novos nomes. O problema pode ser relacionado ao nome da variável ou tabela! Esses casos são raros, mas é melhor prevenir do que remediar. Além disso, será mais fácil para você fornecer um exemplo que use sua situação atual, que é o que mais importa para nós. No caso de ter dados que não deseja mostrar para outros, você pode usar o `ftp` para transferi-lo para `ftp://support.mysql.com/pub/mysql/secret/`. Se os dados são realmente confidenciais, e você não deseja mostrá-los nem mesmo para nós, então vá em frente e providencie um exemplo usando outros nome, mas, por favor considere isso como uma única chance.
- Inclua, se possível, todas as opções fornecidas aos programas relevantes. Por exemplo, indique as opções que você utiliza quando inicializa o daemon `mysqld` e aquelas que são utilizadas para executar qualquer programa cliente MySQL. As opções para programas como o `mysqld` e `mysql`, e para o script `configure`, são frequentemente chaves para respostas e são muito relevantes! Nunca é uma má idéia incluí-las de qualquer forma! Se você usa algum módulo, como Perl ou PHP por favor forneça o número da versão deles também.
- Se sua questão é relacionada ao sistema de privilégios, por favor forneça a saída de `mysqlaccess`, a saída de `mysqladmin reload`, e todas as mensagens de erro que

você obteve quando tentava conectar! Quando você testar seus privilégios, você deve primeiramente executar `mysqlaccess`. Depois, execute `mysqladmin reload version` e tente conectar com o programa que gerou o problema. `mysqlaccess` pode ser encontrado no diretório '`bin`' sob seu diretório de instalação do MySQL.

- Se você tiver um patch para um erro, isso é bom, mas não assuma que o patch é tudo que precisamos, ou que iremos usá-lo, se você não fornecer algumas informações necessárias, como os casos de testes mostrando o erro que seu patch corrige. Nós podemos encontrar problemas com seu patch ou nós podemos não entendê-lo ao todo; se for assim, não podemos usá-lo.

  Se nós não verificarmos exatamente o que o patch quer dizer, nós não poderemos usá-lo. Casos de testes irão ajudar-nos aqui. Mostre que o patch irá cuidar de todas as situações que possam ocorrer. Se nós encontrarmos um caso (mesmo que raro) onde o patch não funcionaria, ele pode ser inútil.
- Palpites sobre qual é o erro, porque ocorre, ou do que ele depende, geralmente estão errados. Mesmo o time MySQL não pode adivinhar antecipadamente tais coisas sem usar um debugger para determinar a causa real do erro.
- Indique na sua mensagem de e-mail que você conferiu o manual de referência e o arquivo de mensagens para que outros saibam que você tentou solucionar o problema.
- Se você obter um `parse error`, por favor confira sua sintaxe com atenção! Se você não conseguiu encontrar nada errado com ela, é extremamente provável que que sua versão corrente do MySQL não suporte a consulta que você está utilizando. Se você estiver usando a versão recente e o manual em `http://www.mysql.com/documentation/manual.php` não cobrir a sintaxe que você estiver usando, o MySQL não suporta sua consulta. Neste caso, suas unicas opções são implementar você mesmo a sintaxe ou enviar uma mensagem para `mysql-licensing@mysql.com` e perguntar por uma oferta para implementá-lo!

  Se o manual cobrir a sintaxe que você estiver usando, mas você tiver uma versão mais antiga do MySQL, você deverá conferir o histórico de alterações do MySQL para ver quando a sintaxe foi implementada. Neste caso, você tem a opção de atualizar para uma nova versão do MySQL. Veja Apêndice D [News], Página 957.
- Se você tiver um problema do tipo que seus dados aparecem corrompidos ou você obtem erros quando você acessa alguma tabela em particular, você deverá primeiro checar depois tentar reparar suas tabelas com `myisamchk` ou `CHECK TABLE` e `REPAIR TABLE`. Veja Cap“ptexi tulo 4 [MySQL Database Administration], Página 207.
- Se você frequentemente obtém tabelas corrompidas, você deve tentar encontrar quando e porque isto acontece! Neste caso, o arquivo '`mysql-data-directory/'hostname'.err`' deve conter algumas informações sobre o que aconteceu. Veja Seção 4.10.1 [Error log], Página 372. Por favor forneça qualquer informação relevante deste arquivo no seu relatório de erro! Normalmente o `mysqld` **NUNCA** deverá danificar uma tabela se nada o finalizou no meio de uma atualização! Se você puder encontrar a causa do fim do `mysqld`, se torna muito mais fácil para nós fornecemos a você uma solução para o problema! Veja Seção A.1 [What is crashing], Página 911.
- Se possível, faça o download e instale a versão mais recente do MySQL para saber se ela resolve ou não o seu problema. Todas versões do MySQL são muito bem testadas

e devem funcionar sem problemas! Acreditamos em deixar tudo, o mais compátivel possível com as versões anteriores, e você conseguirá mudar de versões MySQL em minutos! Veja Seção 2.2.4 [Which version], Página 80.

Se você é um cliente de nosso suporte, por favor envio o seu relatório de erros em `mysql-support@mysql.com` para tratamento de alta prioritário, bem como para a lista de mensagens apropriada para ver se mais alguém teve experiências com (e talvez resolveu) o problema.

Para informações sobre relatar erros no **MyODBC**, veja Seção 12.2.4 [ODBC Problems], Página 873.

Para soluções a alguns problemas comuns, veja Veja Apêndice A [Problems], Página 911.

Quando respostas são enviadas para você individualmente e não para a lista de mensagens, é considerado boa etiqueta resumir as respostas e enviar o resumo para a lista de mensagens para que outras possam ter o benefício das respostas que você recebeu que ajudaram a resolver seu problema!

1.7.1.4 Guia para responder questões na lista de discussão

Se você considerar que sua respota possa ter um amplo interesse, você pode querer postá-la para a lista de mensagens em vez de responder diretamente para a pessoa que perquntou. Tente deixar sua resposta da forma mais genérica possível para que outras pessoas além da que postou a pergunda possam se beneficiar dela. Quando você postar para a lista, por favor tenha certeza que sua resposta não é uma réplica de uma resposta anterior.

Tente resumir a parte essencial da questão na sua resposta, não se sinta obrigado a citar a mensagem original inteira.

Por favor não poste mensagens a partir de seu browser com o modo HTML ligado! Muitos usuários não leem e-mail com browser!

1.7.2 Suporte a Comunidade MySQL Atrvés do IRC (Internet Relay Chat)

Em adição as diversas listas de email, você pode pessoas experientes da comunidade no `IRC` (`Internet Relay Chat`). Estes são os melhores canais atualmente conhecidos por nós:

- **freenode** (veja `http://www.freenode.net/` para servidores)
  - `#mysql` A princípio são questões sobre o MySQL, mas dúvidas sobre outros bancos de dados e SQL são bemvindas.
  - `#mysqlphp` Questões sobre MySQL+PHP, uma combinação popular.
  - `#mysqlperl` Questões sobre MySQL+Perl, outra combinação popular.
- **EFnet** (veja `http://www.efnet.org/` para servidores)
  - `#mysql` Questões sobre MySQL.

Se você está procurando por programas clientes de IRC para conectar a uma rede IRC, dê uma olhada no `X-Chat` (`http://www.xchat.org/`). X-Chat (licença GPL) está disponível para as plataformas Unix e Windows.

1.8 Qual compatibilidade aos padrões o MySQL oferece ?

Esta seção descreve como o MySQL se relaciona aos padrões ANSI/ISO SQL. O Servidor MySQL tem muitas extensões aos padrões SQL, e aqui você descobrirá quais são elas, e como usá-las. Você irá também encontrar informação sobre falta de funcionalidade do Servidor MySQL, e como trabalhar com algumas diferenças.

Nosso objetivo é não restringir, sem um boa razão, a usabilidade do MySQL Server para qualquer uso. Mesmo se não tivermos os recursos para fazer o desenvolvimento para todos os usos possíveis, estamos sempre querendo ajudar e oferecer sugestões para pessoas que estão tentando usar o MySQL Server em novos territórios.

Um dos nossos principais objetivos com o produto é continuar a trabalhar em acordo com o padrão SQL-99, mas sem sacrificar velocidade e confiança. Não estamos receosos em adicionar extensões ao SQL ou suporte para recursos não SQL se ele aumentar extremamente a usabilidade do MySQL Server para uma grande parte de nossos usuários. (A nova interface `HANDLER` no MySQL Server 4.0 é um exeemplo desta estratégia. Veja Seção 6.4.9 [`HANDLER`], Página 595.)

Continuaremos a suportar bancos de dados transacionais e não transacionais para satisfazer tanto o uso pesado na web quanto o uso de missão crítica 24/7.

O MySQL Server foi projetado inicialmente para trabalhar com bancos de dados de tamanho médio (10-100 milhões de registros ou cerca de 100 MB por tabela) em sistemas computacionais pequenos. Continuaremos a extender o MySQL Server para funcionar ainda melhor com banco de dados na ordem de terabytes, assim como tornar possível compilar uma versão reduzida do MySQL mais apropriadas para handhels e uso embutido. O design compacto do servidor MySQL tornam ambas as direções possíveis sem qualquer conflito na árvore fonte.

Atualmente não estamos buscando suporte em tempo real (mesmo se você já puder fazer muitas coisas com nossos serviços de replicação).

Suporte a banco de dados em cluster está planejado para 2004 pela implementação de um novo mecanismo de armazenamento.

Estamos buscando melhoras no fornecimento de suporte a XML no servidor de banco de dados.

1.8.1 Qual Padrão o MySQL Segue?

Entry-level SQL-92. ODBC levels 0-3.51.

We are aiming toward supporting the full SQL-99 standard, but without concessions to speed and quality of the code.

1.8.2 Executando o MySQL no modo ANSI

Se você inicializa o `mysqld` com a opção `--ansi` ou `--sql-mode=ANSI`, o seguinte comportamento é alterado no MySQL:

- `||` é um oprador de concatenação de strings em vez de um sinônimo para `OR`.

- '"' é tratado como um caracter identificados (com o caracter de aspasr '`' do MySQL Server)e não um caracter de string. Você ainda pode usar '`' para citar identificadores no modo ANSI. Uma implicação disto é que você não pode usar aspas duplas para citar um string literal, porque ela será intepretada como um identificador.
- Você pode ter qualquer número de espaços entre um nome de função e o '('. Isto faz com que todos nomes de funções sejam tratadas como palavras reservadas. Como resultado, se você quiser acessar qualquer banco de dados, tabelas ou coluna que é uma palavra reservada, você deve colocá-lo entre aspas. Por exemplo, por haver a função `USER()`, o nome da tabela `user` no banco de dados `mysql` e a coluna `User` nesta tabela se torna reservada, assim você deve colocá-la entre aspas:

```
SELECT "User" FROM mysql."user";
```

- `REAL` é um sinônimo para `FLOAT` no lugar de um sinônimo de `DOUBLE`.
- O nível de isolamento padrão de um transação é `SERIALIZABLE`. Veja Seção 6.7.6 [`SET TRANSACTION`], Página 619.
- Você pode usar um campo/expressão em `GROUP BY` que não está na lista de campos.

Executando o servidor em modo ANSI é o mesmo que iniciá-lo com estas opções:

```
--sql-mode=REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,
IGNORE_SPACE,ONLY_FULL_GROUP_BY --transaction-isolation=serializable
```

No MySQL 4.1, você pode conseguir o mesmo efeito com estas duas instruções:

```
SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET GLOBAL sql_mode=
  "REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ONLY_FULL_GROUP_BY";
```

No MySQL 4.1.1 a última opção `sql_mode` também pode ser dada com:

```
SET GLOBAL sql_mode="ansi";
```

No caso acima o `sql_mode` estará configurado com todas as opções que são relevantes para o modo ANSI. Você pode verificar o resultado fazendo:

```
mysql> SET GLOBAL sql_mode="ansi";
mysql> SELECT @@GLOBAL.sql_mode;
        -> "REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ONLY_FULL_GROUP
```

1.8.3 Extensões do MySQL para o Padrão SQL-92

O MySQL fornece algumas extensões que você provavelmente não irá encontrar em alguns bancos de dados SQL. Fique avisado que se você usá-las, seu código pode não ser mais portável para outros servidores SQL. Em alguns casos, você pode escrever código que inclui extensões MySQL, mas continua portável, usando comentários da forma `/*! ...*/`. Neste caso, o MySQL irá analisar e executar o código com o comentário como irá fazer com qualquer outra instrução MySQL, mas outros servidores SQL irão ignorar as extensões. Por exemplo:

```
SELECT /*! STRAIGHT_JOIN */ nome_campo FROM table1,table2 WHERE ...
```

Se você adicionar um número de versão depois do `'!'`, a sintaxe só será executada se a versão do MySQL é igual ou maior que o número de versão usado:

```
CREATE /*!32302 TEMPORARY */ TABLE t (a INT);
```

O exemplo acima significa que se você tiver uma versão do MySQL 3.23.02 ou mais nova, então o MySQL irá usar a palavra-chave `TEMPORARY`

Extensões MySQL são listadas abaixo:

- Os tipos de campo `MEDIUMINT`, `SET`, `ENUM` e os diferentes tipos `BLOB` e `TEXT`.
- Os atributos de campos `AUTO_INCREMENT`, `BINARY`, `NULL`, `UNSIGNED` e `ZEROFILL`.
- Todas comparações de strings por padrão são caso insensitivo, com classificação ordenada determinada pelo conjunto de caracteres corrente (ISO-8859-1 Latin1 por padrão). Se você não gosta disso você deverá declarar suas colunas com o atributo `BINARY` ou usar o operador `BINARY`, que fazendo com que as comparações sejam feitas de acordo com a ordem ASCII usada na máquina servidora do MySQL.
- O MySQL mapeia cada banco de dados em um diretório sob o diretório de dados do MySQL, e tabelas internamente num banco de dados para arquivos no diretório do banco de dados.

  Isto tem algumas implicações:

  – Nomes de bancos de dados e tabelas são caso sensitivoo no MySQL em sistemas operacionais que possuem o sistema de arquivos caso sensitivoo (como na maioria dos sistemas Unix). Veja Seção 6.1.3 [Name case sensitivity], Página 473.

  – Nomes de Bancos de dados, tabelas, índices, campos ou apelidos pode começar com um dígito (porém não podem consistir somente de digitos).

  – Você pode usar comandos padrão do sistemas para fazer backups, renomear, apagar e copiar tabelas. Por exemplo, para renomear uma tabela, renomeie os arquivos '`.MYD`', '`.MYI`' e '`.frm`'. para o nome da tabela correspondente.
- Em algumas instruções SQL, você pode acessar tabelas de diferentes bancos de dados com a sintaxe `nome_bd.nome_tbl`. Alguns servidores SQL fornecem a mesma funcionalidade mas chamam isto de `User space`. O MySQL não suporta tablespaces como em: `create table ralph.my_table...IN minha_tablespace`.
- `LIKE` é permitido em campos numéricos.
- O uso de `INTO OUTFILE` e `STRAIGHT_JOIN` em uma instrução `SELECT`. Veja Seção 6.4.1 [SELECT], Página 562.
- A opção `SQL_SMALL_RESULT` em uma instrução `SELECT`.
- `EXPLAIN SELECT` para obter uma descrição de como as tabelas são ligadas.
- A utilização de nomes de índices, índices em um prefixo de um campo, e uso de `INDEX` ou `KEY` em uma instrução `CREATE TABLE`. Veja Seção 6.5.3 [CREATE TABLE], Página 598.
- O uso de `TEMPORARY` ou `IF NOT EXISTS` com `CREATE TABLE`.
- O uso de `COUNT(DISTINCT lista)` onde 'lista' é maior que um elemento.
- O uso de `CHANGE nome_campo`, `DROP nome_campo`, ou `DROP INDEX`, `IGNORE` ou `RENAME` em uma instrução `ALTER TABLE`. Veja Seção 6.5.4 [ALTER TABLE], Página 608.
- O uso de `RENAME TABLE`. Veja Seção 6.5.5 [RENAME TABLE], Página 612.
- Utilização de múltiplas cláusulas `ADD`, `ALTER`, `DROP`, ou `CHANGE` em uma instrução `ALTER TABLE`.
- O uso de `DROP TABLE` com as palavras-chave `IF EXISTS`.

- Você pode remover múltiplas tabelas com uma instrução única `DROP TABLE`.
- As cláusulas `ORDER BY` e `LIMIT` das instruções `UPDATE` e `DELETE`.
- Sintaxe `INSERT INTO ... SET col_name = ....`
- A cláusula `DELAYED` das instruções `INSERT` e `REPLACE`.
- A cláusula `LOW_PRIORITY` das instruções `INSERT`, `REPLACE`, `DELETE` e `UPDATE`.
- O uso de `LOAD DATA INFILE`. Em alguns casos essa sintaxe é compatível com o Oracle `LOAD DATA INFILE`. Veja Seção 6.4.8 [`LOAD DATA`], Página 588.
- As intruções `ANALYZE TABLE`, `CHECK TABLE`, `OPTIMIZE TABLE`, e `REPAIR TABLE`.
- A instrução `SHOW`. Veja Seção 4.6.8 [`SHOW`], Página 302.
- Strings podem ser fechadas pelo '`"`' ou '`'`', não apenas pelo '`'`'.
- O uso do meta-caractere de escape '`\`'.
- A instrução `SET OPTION`. Veja Seção 5.5.6 [`SET OPTION`], Página 460.
- Você não precisa nomear todos os campos selecionados na parte `GROUP BY`. Isto fornece melhor performance para algumas consultas específicas, mas muito comuns. Veja Seção 6.3.7 [Group by functions and modifiers], Página 555.
- Pode ser especificado `ASC` e `DESC` com o `GROUP BY`.
- Para tornar mais fácil para usuários que venham de outros ambientes SQL, o MySQL suporta apelidos (aliases) para várias funções. Por exemplo, todas funções de string suportam as sintaxes ANSI SQL e ODBC.
- O MySQL entende os operadores `||` e `&&` como ou(OR) e e(AND) logicos, como na linguagem de programação C. No MySQL, `||` e `OR` são sinônimos, assim como `&&` e `AND`. Devido a esta ótima sintaxe, o MySQL não suporta o operador ANSI SQL para concatenação de strings `||`; em vez disso, use o `CONCAT()`. Como `CONCAT()` aceita vários argumentos, é fácil converter o uso do operador `||` para MySQL.
- `CREATE DATABASE` or `DROP DATABASE`. Veja Seção 6.5.1 [`CREATE DATABASE`], Página 597.
- O operador `%` é um sinônimo para `MOD()`. Isto é, `N % M` é equivalente a `MOD(N,M)`. `%` é suportado para programadores C e para compatibilidade com o PostgreSQL.
- Os operadores `=`, `<>`, `<=` ,`<`, `>=`,`>`, `<<`, `>>`, `<=>`, `AND`, `OR` ou `LIKE` podem ser utilizados em comparações de campos a esquerda do `FROM` nas instruções `SELECT`. Por exemplo:

```
mysql> SELECT col1=1 AND col2=2 FROM nome_tabela;
```

- A função `LAST_INSERT_ID()`. Veja Seção 12.1.3.32 [`mysql_insert_id()`], Página 803.
- Os operadores extendidos `REGEXP` e `NOT REGEXP` utilizados em expressões regulares.
- `CONCAT()` ou `CHAR()` com um ou mais de dois argumentos. (No MySQL, estas funções receber qualquer número de argumentos.)
- As funções `BIT_COUNT()`, `CASE`, `ELT()`, `FROM_DAYS()`, `FORMAT()`, `IF()`, `PASSWORD()`, `ENCRYPT()`, `MD5()`, `ENCODE()`, `DECODE()`, `PERIOD_ADD()`, `PERIOD_DIFF()`, `TO_DAYS()` ou `WEEKDAY()`.
- Uso de `TRIM()` para cortar substrings. o SQL-99 só suporta remoção de caracteres únicos.
- As funções do `GROUP BY`: `STD()`, `BIT_OR()`, `BIT_AND()` e `BIT_XOR()` e `GROUP_CONCAT()`. Veja Seção 6.3.7 [Group by functions and modifiers], Página 555.

- Uso de `REPLACE` no lugar de `DELETE + INSERT`. Veja Seção 6.4.7 [`REPLACE`], Página 587.
- As instruções `FLUSH`, `RESET` e `DO`.
- A possibilidade de configurar variáveis em uma instrução com `:=`:

```
SELECT @a:=SUM(total),@b=COUNT(*),@a/@b AS media FROM tabela_teste;
SELECT @t1:=(@t2:=1)+@t3:=4,@t1,@t2,@t3;
```

1.8.4 Diferenças do MySQL em Comparação com o SQL-92

Nós tentamos fazer com que o MySQL siguisse os padrões ANSI SQL (SQL-92/SQL-99) e o ODBC SQL, mas em alguns casos, o MySQL realiza operações de forma diferente:

- Para campos `VARCHAR`, expaços extras são removidos quando o valor é armazenado. Veja Seção 1.8.6 [Bugs], Página 54.
- Em alguns casos, campos `CHAR` são alterados sem perguntas para o tipo de campo `VARCHAR`. Veja Seção 6.5.3.1 [Silent column changes], Página 607.
- Privilégios para uma tabela não são negadas automaticamente quando você apaga uma tabela. Você deve usar explicitamente um `REVOKE` para negar privilégios para uma tabela. Veja Seção 4.4.1 [`GRANT`], Página 254.

Para uma lista priorizada indicando quando novas extensões serão adicionadas ao MySQL você deve consultar lista TODO online do MySQL em `http://www.mysql.com/doc/en/TODO.html`. Esta é a última versão da lista TODO neste manual. Veja Seção 1.6 [TODO], Página 27.

1.8.4.1 Subqueries

MySQL Version 4.1 supports subqueries and derived tables (unnamed views). Veja Seção 6.4.2 [Subqueries], Página 570.

For MySQL versions prior to 4.1, most subqueries can be successfully rewritten using joins and and other methods. Veja Seção 6.4.2.11 [Rewriting subqueries], Página 578.

1.8.4.2 `SELECT INTO TABLE`

O MySQL ainda não suporta a extensão SQL do Sybase: `SELECT ... INTO TABLE ...`. MySQL suporta a sintaxe ANSI SQL `INSERT INTO ... SELECT ...`, que é basicamente a mesma coisa. Veja Seção 6.4.3.1 [INSERT SELECT], Página 581.

```
INSERT INTO tblTemp2 (fldID)
       SELECT tblTemp1.fldOrder_ID
       FROM tblTemp1 WHERE tblTemp1.fldOrder_ID > 100;
```

De maneira alternativa, você pode usar `SELECT INTO OUTFILE...` ou `CREATE TABLE ... SELECT` para resolver seu problema.

1.8.4.3 Transações e Operações Atômicas

O MySQL Server (versão 3.23-max e todas as versões 4.0 e acima) suportam transações com os `mecanismos de armazenamento transacionais InnoDB` e `BDB`. `InnoDB` fornece compatibilidade *total* com `ACID`. Veja Cap“ptexi tulo 7 [Tipos de tabelas], Página 630.

Os outros tipos de tabelas não transacionais (tais como `MyISAM`) no MySQL Server seguem um paradigma diferente para integridade de dados chamado "`Operções Atômicas`." Em termos de transação, tabelas `MyISAM` efetivamente sempre operam em modo `AUTOCOMMIT=1`. Operações atômicas geralmente oferecem integridade comparável com a mais alta performance.

Com o MySQL Server suportando ambos os paradigmas, o usuário pode decidir se precisa da velocidade das operações atômicas ou se precisa usar recursos transacionais em seu aplicativo. Esta escolha pode ser feita em uma base por tabela.

Como notado, a comparação para tabelas transacionais vs. não transacionais As noted, the trade off for transactional vs. non-transactional table se encontra em grande parte no desempenho. Tabelas transacionais tem uma exigência de memória e espaço em disco significantemente maior e maior sobrecarga da CPU. Tipos de tabelas transacionais como `InnoDB` oferecem muitos recursos únicos. O projeto modular do MySQL Server permite o uso concorrente de todas estes mecanismos de armazenamento para servir a diferentes exigências e oferecer um ótimo desempenho em todas as situações.

Mas como fazer uso dos recursos do MySQL Server para manter uma integridade rigorosa mesmo com tabelas `MyISAM` não transacionais e como este recurso se compara com os tipos de tabelas transacionais?

1. No paradigma transacional, se as suas aplicações são escritas de uma forma que é dependente na chamada de `ROLLBACK` em vez de `COMMIT` em situações críticas, então transações são mais convenientes. Além disso, transações asseguram que atualizações inacabadas ou atividades corrompidas não sejam executadas no banco de dados; o servidor oferece uma oportunidade para fazer um rollback automático e seu banco de dados é mantido.

   O MySQL Server, na maioria dos casos, permite a você resolver potenciais problemas incluindo simples conferências antes das atualizações e executando scripts simples que conferem inconsistências no banco de dados e, automaticamente, repara ou avisa caso isto ocorra. Perceba que apenas usando o log do MySQL ou mesmo adicionando um log extra, pode-se corrigir tabelas perfeitamente sem nenhuma perda de integridade.

2. Mais do que nunco, atualizações transacionais fatais podem ser reescritas para serem atômicas. De fato podemos dizer que todos problemas de integridade que transações resolvem podem ser feitas com `LOCK TABLES` ou atualizações atômicas, assegurando que você nunca irá ter uma finalização automática da tabela, o que é um problema comum em bancos de dados transacionais.

3. Nem mesmo transações podem prevenir todas as falhas se o servidor cair. Nestes casos mesmo um sistema transacional pode perder dados. A diferença entre sistemas diferentes é apenas em quão pequeno é o lapso de tempo em que eles podem perder dados. Nenhum sistema é 100% seguro, somente "seguro o suficiente." Mesmo o Oracle, com reputação de ser o mais seguro bancos de dados transacionais, tem relatos de algumas vezes perder dados nestas situações.

   Para estar seguro com o MySQL Server, você apenas deve fazer backups e ter o log de atualizações ligado. Com isto você pode se recuperar de qualquer situação possível com bancos de dados transacionais. É sempre bom ter backups, independente de qual banco de dados você usa.

O paradigma transacional tem seus benefícios e suas desvantagens. Muitos usuários e desenvolvedores de aplicações dependem da facilidade com a qual eles podem codificar contornando problemas onde abortar parece ser, ou é necessário. No entanto, se você é novo no paradigma de operações atômicas ou tem mais familiaridade ou conforto com transações, considere o benefício da velocidade que as tabelas não transacionais podem oferece, na ordem de 3 a 5 vezes da velocidade que as tabelas transacionais mais rápidas e otimizadas.

Em situações onde integridade é de grande importância, as atuais características do MySQL permitem níveis transacionais ou melhor confiança e integridade. Se você bloquear tabelas com `LOCK TABLES` todos as atualizações irão ser adiadas até qualquer verificação de integridade ser feita. Se você só obter um bloqueio de leitura (oposto ao bloqueio de escrita), então leituras e inserções poderão ocorrer. Os novos registros inseridos não poderão ser visualizados por nenhum dos clientes que tiverem um bloqueio de `LEITURA` até eles liberarem estes bloqueios. Com `INSERT DELAYED` você pode enfileirar inserções em uma fila local, até os bloqueios serem liberados, sem que o cliente precise esperar atá a inserção completar. Veja Seção 6.4.3.2 [INSERT DELAYED], Página 582.

"Atômico", no sentido em que nós mencionamos, não é mágico. Significa apenas que você pode estar certo que enquanto cada atualização específica está sendo executada, nenhum outro usuário pode interferir com ela, e nunca haverá um rollback automático (que pode acontecer em sistemas baseados em transações se você não tiver muito cuidado). O MySQL também assegura que nunca ocorrerá uma leitura suja.

A seguir estão algumas técnicas para trabalhar com tabelas não transacionais:

- Loops que precisam de transações normalmente pode ser codificados com a ajuda de `LOCK TABLES`, e você não precisa de cursores para atualizar regitros imeditamente.
- Para evitar o uso do `ROLLBACK`, você pode usar as seguintes estratégias:
  1. Use `LOCK TABLES ...` para fazer um lock todas as tabelas que você quer acessar.
  2. Condições de teste.
  3. Atualize se estiver tudo OK.
  4. Use `UNLOCK TABLES` para liberar seus locks.

  Isto é normalmente um método muito mais rápido que usar transações com possíveis `ROLLBACK`s, mas nem sempre. A única situação que esta solução não pode tratar é quando alguém mata a threads no meio de uma atualização. Neste caso, todas os locks serão liberados mas algumas das atualização podem não ter sido execuadas.
- Você também pode usar funções para atualizar registros em uma única operação. Você pode conseguir uma aplicação muito eficiente usando as seguintes técnicas:
  - Modifique campos em relação ao seus valores atuais.
  - Atualize apenas aqueles campos que realmente tiveram alterações.

  Por exemplo, quando fazemos atualizações em alguma informação de cliente, atualizamoa apenas os dados do clientes que alteraram e testamos apenas aqueles com dados alterados ou dados que dependem dos dados alterados, mudaram em comparação com o registro original. O teste dos dados alterados é feito com a cláusula `WHERE` na instrução `UPDATE`. Se o registro não foi atualizado, mandamos ao cliente uma mensagem: "Some of the data you have changed has been changed by another user." Então mostramos o

registro antigo versus o novo em uma janela, assim o usuário pode decidir qual versão do registro de cliente de ser usado.

Isto nos dá algo similar a lock de colunas mas que, na verdade, é melhor porque apenas atualizamos algumas das colunas, usando valores relativos ao seu valor atual. Isto significa que instruções `UPDATE` comuns se parecem com estas:

```
UPDATE nometabela SET pay_back=pay_back+125;

UPDATE customer
  SET
    customer_date='current_date',
    address='new address',
    phone='new phone',
    money_he_owes_us=money_he_owes_us-125
  WHERE
    customer_id=id AND address='old address' AND phone='old phone';
```

Como você pode ver, isto é muito eficiente e funciona mesmo se outro cliente alterar os valores nas colunas `pay_back` ou `money_he_owes_us`.

- Em muitos casos, usuários querem fazer `ROLLBACK` e/ou `LOCK TABLES` com o propósito de gerenciarem identificadores únicos para algumas tabelas. Isto pode ser tratado muito mais eficientemente usando uma coluna `AUTO_INCREMENT` e também uma função SQL `LAST_INSERT_ID()` ou a função da API C `mysql_insert_id()`. Veja Seção 12.1.3.32 [`mysql_insert_id()`], Página 803.

  Geralmente você pode codificar evitando lock de registro. Algumas situações realmente precisam disto, e tabelas `InnoDB` suportam lock de regitstro. Comoo MyISAM, você pode usar uma coluna de flag na tabela e fazer algo como a seguir:

  ```
  UPDATE nome_tbl SET row_flag=1 WHERE id=ID;
  ```

  O MySQL retorna 1 para o número de linhas afetadas se as linhas foram encontradas e `row_flag` já não era 1 na linha original.

  Você pode pensar nisto como se o MySQL Server tivesse alterado a consulta anterior para:

  ```
  UPDATE nome_tbl SET row_flag=1 WHERE id=ID AND row_flag <> 1;
  ```

1.8.4.4 Stored Procedures e Triggers

Steored procedures estão sendo implementadas em nossa versão 5.0 na árvore de desenvolvimento. Veja Seção 2.3.4 [Instalando da árvore de fontes], Página 100.

Este esforço é baseado no SQL-99, que têm uma sintaxe básica similar (mas não idêntica) ao Oracle PL/SQL. Em adição a isto, estamoas implementando o framework SQL-99 enganchar em linguagens externas.

Uma Stored Procedure é um conjunto de comandos SQL que podem ser compilados e armazenados no servidor. Uma fez feito isso, os clientes não necessitam reescrever toda a consulta mas podem fazer referência à stored procedure. Isto fornece melhor performance porque a query necessita ser analisada pelo servidor somente uma vez, e necessita menos informação para ser enviada entre o servidor e o cliente. Você também pode elevar o

nível conceitual tendo bibliotecas de funções no servidor. No entanto, stored procedures aumentam a carga no servidor de banco de dados, já que grande parte do trabalho é feito do lado do servidor e menos do lado do cliente (aplicação).

Triggers estão programados para serem implementados no MySQL versão 5.1. Um trigger é um tipo de stored procedure que é chamado quando um evento em particular ocorre. Por exemplo, você poderia configurar uma stored procedure que é disparada toda vez que um registro for apagado de uma tabela transacional que automaticamente apaga o cliente correspondente de uma tabela de clientes quando todas as transações forem removidas.

1.8.4.5 Chaves Estrangeiras

No MySQL Server 3.23.44 e posterior, tabelas `InnoDB` suportam verificação de restrição de chaves estrangeiras, incluindo `CASCADE`, `ON DELETE`, e `ON UPDATE`. Veja Seção 7.5.5.2 [Restrições de chaves estrangeiras do InnoDB], Página 654.

Para outros tipos de tabela, o MySQL Server atualmente apenas analisa a sintaxe de `FOREIGN KEY` no comando `CREATE TABLE`, mas não usa/armazena esta informação. Em um futuro próximo esta implementação será estendida para que assim a informação seja armazenada num arquivo de especificação de tabela e possa ser recuperado por `mysqldump` e ODBC. Em um estágio posterior, restrições de chaves estrangeiras serão implementadas para tabelas `MyISAM`.

Note que as chaves estrangeiras no SQL não são usadas para ligar tabelas, mas são usadas para verificar a integridade referencial. Se você deseja obter resultados de múltiplas tabelas de uma instrução `SELECT`, você pode fazer isto ligando tabelas:

```
SELECT * FROM table1,table2 WHERE table1.id = table2.id;
```

Veja Seção 6.4.1.1 [`JOIN`], Página 567. Veja Seção 3.6.6 [Exemplos de chaves estrangeiras], Página 198.

Quando usada como uma restrição, `FOREIGN KEY`s não precisa ser usado se a aplicação insere duas linhas em tabelas `MyISAM` na ordem apropriada.

Para tabelas `MyISAM`, você pode contornar a falta de `ON DELETE` adicionando a instrução `DELETE` apropriada a uma aplicação quando você deletar registros de uma tabela que tem uma chave estrangeira. Na prática isto é mais rápido e muito mais portável que utilizar chaves estrangeiras.

No MySQL Server 4.0 você pode utilizar deleções multi-tabela para apagar linha de muitas tabelas com um comando. Veja Seção 6.4.5 [DELETE], Página 585.

A sintaxe `FOREIGN KEY` sem `ON DELETE ...` é usada geralmente por aplicacões ODBC para produzir cláusulas `WHERE` automáticas.

Note que chaves estrangeiras são mal usadas com frequência, o que pode causar graves problemas. Mesmo quando usado apropriadamente, o suporte a chaves estrangeiras não é uma solução mágica para o problema de integridade referêncial, embora possa ajudar.

Algumas vantagens das chaves estrangeiras:

- Assumindo o projeto apropriado das relações, as restrições de chaves estrangeiras tornarão mais difícil para um programador introduzir uma inconsistência no banco de dados.
- Usar atualizações e deleções em cascata pode simplificar o código do cliente.

- Regras de chaves estrangeiras projetados apropriadamente ajudam ao documentar a relação entre as tabelas.

Desvantagens:

- Erros, que são facéis de se ter ao projetar a relação das chaves, podem causar graves problemaspor exemplo, regras circulares ou a combinação errada de uma deleção em cascata.
- Verificação adicional no banco de dados afeta o desempenho, por esta razão algumas das principais aplicações comerciais codificam sua lógica no nível da aplicação.
- Não é incomum para um DBA fazer uma topologia complexa de relações que torna muito difícl, e em alguns casos impossível, fazer backup ou restaurar tabelas individuais.

1.8.4.6 Views

Views estão senda implementadas atualmente e aparecerão na versão 5.0 e 5.1 do MySQL Server.

Historicamente o MySQL Server tem sido mais usado em aplicações e sistemas web onde o desenvolvedor da aplicação tem total controle sobre o uso do banco de dados. É claro que o uso aumentou em várias vezes e então descobrimos que um crescente números de usuários consideram views como um importante aspecto.

Unnamed views (*derived tables*, uma seubquery na cláusula `FROM` de uma `SELECT`) já estão implementadas na versão 4.1.

Views geralmente são muito úteis para permitir aos usuários acessar uma série de relações (tabelas) como uma tabela, e limitar o acesso a apenas estas relações. Views também podem ser usadas para restringir o acesso aos registros (um subconjunto de uma tabela em particular). Mas views não são necessárias para restringir o acesso a registros já que o MySQL Server tem um sofisticado sistema de privilégios. Veja Seção 4.3 [Sistema de privilégios], Página 226.

Muitos SGBD não permitem atualizar nenhum registro em uma view, mas você tem que fazer as atualizações em tabelas separadas.

Em nosso projeto de implemtação de views, nós buscamos (tanto quanto for possível dentro do SQL) compatibilidade com "**Codd's Rule #6**" para sistemas de banco de dados relacionais: todos os views que são teoricamente atualizáveis, devem se atualizados também na prática.

1.8.4.7 '`--`' como Início de Comentário

Outros bancos de dados SQL usam '`--`' para iniciar comentários. O MySQL usa '`#`' como o caractere para início de comentário, mesmo se a ferramenta de linha de comando `mysql` remover todas linhas que começam com '`--`'. Você também pode usar o comentário no estilo C `/*isto é um comentário*/` com o MySQL Server. Veja Seção 6.1.6 [Comentários], Página 479.

O MySQL Server versão 3.23.3 e superior suporta o estilo de comentário '`--`' somente se o comentário for seguido por um caractere de espaço (ou por um caracter de controle como uma nova linha). Isto ocorre porque este estilo de comentário causou muitos problemas

com queries SQL geradas automaticamente que usavam algo como o código seguinte, onde automaticamente erá inserido o valor do pagamento para `!pagamento!`:

```
UPDATE nome_tabela SET credito=credito-!pagamento!
```

O que você acha que irá acontecer quando o valor de `pagamento` for negativo? Como `1--1` é legal no SQL, nós achamos terrível que '`--`' signifique início de comentário.

Usando a nossa implementação deste método de comentário no MySQL Server Version 3.23.3 e posterior, `1-- Isto é um comentário` é atualmente seguro.

Outro recurso seguro é que o cliente de linha de comando `mysql` remove todas as linhas que iniciam com '`--`'.

A seguinte discussão somente interessa se você estiver executando uma versão do MySQL inferior a versão 3.23:

Se você tem um programa SQL em um arquivo texto que contêm comentários '`--`' você deverá usar:

```
shell> replace " --" " #" < arquivo-texto-com-comentário.sql \
         | mysql banco-de-dados
```

No lugar de:

```
shell> mysql banco-de-dados < arquivo-texto-com-comentario.sql
```

Você também pode editar o próprio arquivo de comandos alterando os comentários '`--`' para '`#`':

```
shell> replace " --" " #" -- arquivo-texto-com-comentario.sql
```

Desfaça utilizando este comando:

```
shell> replace " #" " --" -- arquivo-texto-com-comentario.sql
```

1.8.5 Como o MySQL Lida com Restrições

Como o MySQL lhe permite trabalhar com tabelas transacionais e não transacionais (que não permitem rollback), o tratamento de restrições é um pouco diferente no MySQL que em outros bancos de dados.

Temos que tratar o caso quando você atualiza diversos registros com uma tabela não transacional que não pode fazer rollback em erros.

A filosofia básica é tentar obter um erro para qualquer coisa que possamos detectar em temp de compilação mas tentar recuperar de qualquer erro que abtemos em tempo de execução. Fazemos isto na maiorioa dos casos, mas não para todos ainda. Veja Seção 1.6.4 [TODO future], Página 29.

A opção básica que o MySQL tem é parar a instrução no meio ou fazer o melhor para se recuperar do problema e continuar.

A seguir mostramos o que acontece com diferentes tipos de restrições.

1.8.5.1 Restrições de PRIMARY KEY / UNIQUE

Normalmente você receberá um erro quando tentar fazer um `INSERT` / `UPDATE` de um registro que cause uma violação de uma chave primária, chave única ou chave estrangeira. Se você

estiver usando um mecanismo de armazenamento transacional, como InnoDB, o MySQL automaticamente fará um rollback da transação. Se você estiver usando mecanismos de armazenemento não transacionais o MySQL irá para no registro errado e deiar o resto dos registros se processamento.

Para tornar a vida mais fácil o MySQL adicionou suporte a diretiva `IGNORE` para a maioria dos comandos que podem causar uma violação de chave (como `INSERT IGNORE ...`). Neste caso o MySQL irá ignorar qualquer violação de chave e continuará com o processamento do próximo registro. Você pode obter informação sobre o que o MySQL fez com a função da API `mysql_info()` API function e em versões posteriores do MySQL 4.1 com o comando `SHOW WARNINGS`. Veja Seção 12.1.3.30 [mysql_info], Página 802. Veja Seção 4.6.8.9 [SHOW WARNINGS], Página 323.

Note que no momento apenas as tabelas `InnoDB` suportam chaves estrangeiras. Veja Seção 7.5.5.2 [Restrições de chaves estrangeiras no InnoDB], Página 654.

O suporte a chaves estrangeiras nas tabelas `MyISAM` está programado para ser incluída na arvoré de fonte do MySQL 5.0.

1.8.5.2 Restrições de `NOT NULL`

Para poder suportar um fácil tratamento de tabelas não transacionais todos os campos no MySQL têm valores padrão.

Se você inserir um valor 'errado' em uma coluna como um `NULL` em uma coluna `NOT NULL` ou um valor numérico muito grande em um campo numérico, o MySQL irá atribuir a coluna o 'melhor valor possível' em vez de dar uma mensagem de erro. Para strings este valor é uma string vazia ou a maior string possível que possa estar na coluna.

Isto significa que se você tentar armazenar `NULL` em uma coluna que não aceita valores `NULL`, o MySQL Server armazenará 0 ou `''` (strig vazia) nela. Este último comportamento pode, para uma simples inserção de registro, ser alterado com a opção de compilação `-DDONT_USE_DEFAULT_FIELDS`.) Veja Seção 2.3.3 [Opções de configuração], Página 98. Isto faz com que as instruções `INSERT` gerem um erro a menos que você explicite valores específicos para todas as colunas que exigem um valor diferente de `NULL`.

A razão para as regras acima é que não podemos verificar estas condições antes da consulta começar a executar. Se encontrarmos um problema depois de atualizar algumas linahs, não podemos fazer um rollback já que o tipo de tabela não suporta isto. A opção de parar não é tão boa como no caso em que a atualização esteja feita pela metade que é provavelmente o pior cenário possível. Neste caso é melhor 'fazer o possível' e então continuar como se nada tivesse acontecido. No MySQL 5.0 plenejamos melhorar into forncendo avisos para conversões automáticas de campo, mais uma opção para deixar você fazer um rollback das instruções que usam apenas tabelas transacionais no caso de tal instrução fizer uma definição de campo não permitida.

O mostrado acima significa que não se deve usar o MySQL para verificar o conteúdo dos campos, mas deve se fazê-lo por meio da aplicação.

1.8.5.3 Restrições de ENUM e SET

No MySQL 4.x ENUM não é uma restrição real, mas um modo mauis eficiente de armazenar campos que possam apenas conter um conjunto de valores dados. Isto é devido as mesmas razões pelas quais NOT NULL não é respeitado. Veja Seção 1.8.5.2 [restrições NOT NULL], Página 53.

Se você inserir um valor errado em um campo ENUM, ele será configurado com uma string vazia em um contexto string. Veja Seção 6.2.3.3 [ENUM], Página 499.

Se você inserir uma opção errada em um campo SET, o valor errado será ignorado. Veja Seção 6.2.3.4 [SET], Página 500.

1.8.6 Erros Conhecidos e Deficiências de Projetos no MySQL

1.8.6.1 Erros da Versão 3.23 Corrigidos em Versões Posteriores do MySQL

Os seguintes erros/bugs conhecidos não estão corrigidos no MySQL 3.23 porque corrigí-los involveria a mudança de muito código, o que poderia introduzir outros erros, talvez piores. Os erros são também classificados como 'não fatal' ou 'tolerável'.

- Pode se obter um deadlock ao fazer LOCK TABLE em multiplas tabelas e então na mesma conexão fizer um DROP TABLE em uma delas enquanto outra thread está tentando bloquear a tabela. Pode-se no entanto fazer um KILL em qualquer uma das threads envolvidas para resolver isto. Corrigido na versão 4.0.12
- SELECT MAX(campo_chave) FROM t1,t2,t3... onde uma das três tabelas está vazia não retorna NULL, mas sim o valor máximo da coluna. Corrigido na versão 4.0.11.
- DELETE FROM heap_table sem um WHERE não funcionam em tabelas HEAP com lock.

1.8.6.2 Open Bugs / Deficiências de Projeto no MySQL

Os seguintes problemas são conhecidos e tem prioridade muito alta para serem corrigidos:

- FLUSH TABLES WITH READ LOCK não bloqueia CREATE TABLE ou COMMIT, que pode criar um problema com a posição do log binário ao se fazer um backup completo de tabelas e do log binário.
- ANALYZE TABLE em uma tabela BDB pode, em alguns, casos inutilizar a tabela até que se reinicie o servidor mysqld. Quando isto acontecer você irá ver o seguinte tipo de erro no arquivo de erros do MySQL.

```
001207 22:07:56  bdb:  log_flush: LSN past current end-of-log
```

- O MySQL aceita parenteses na parte FROM, mas os ignora sem aviso. A razão pela qual não são retornados erros é que muitos clientes que geram consultas automaticamente adicionam parentesis na parte FROM mesmo onde eles não são necessários.
- Concatenar muitos RIGHT JOINS ou combinar joins LEFT e RIGHT na mesma consulta podem dar uma resposta incorreta ja que o MySQL só gera registros NULL para tabelas que precedem um join LEFT ou antes de um join RIGHT. Isto será corrigido na versão 5.0 junto com o suporte a parentesis na parte FROM.

- Não execute `ALTER TABLE` em uma tabela `BDB` em que você estiver executando transações multi-instruções não completadas. (A transação provavelmente será ignorada).
- `ANALYZE TABLE`, `OPTIMIZE TABLE` e `REPAIR TABLE` podem causar problemas em tabelas para as quais você estiver usando `INSERT DELAYED`.
- Fazendo um `LOCK TABLE ..` e `FLUSH TABLES ..` não garante que não existem transações não terminadas em progresso na tabela.
- Tabelas BDB são um pouco lentas para abrir. Se você tiver várias tabelas BDB em um banco de dados, gastará muito tempo para usar o cliente `mysql` no banco de dados se você não estiver usando a opção `-A` ou se você estiver usando `rehash`. Isto é percebido principalmente quando você tiver um cache de tabelas grandes.
- A replicação utiliza o log a nivel de consulta: o master grava a consulta no log binário. Isto é um rápido, compacto e eficiente método de registro o que funciona perfeitamente na maioria dos casos. Embora nunca tenhamos ouvido sobre um caso ocorrido, há uma chance teórica que o dado no master e slave sejam diferente se uma consulta é feita de tal modo que a modificação do dado é não determinística, isto é, deixar ao desejo do otimizador de consultas (o que geralmente não é uma boa prática, mesmo fora da replicação!). Por exemplo:
  - `CREATE ... SELECT` ou `INSERT ... SELECT` que preenchem com zeros ou `NULL` uma coluna `auto_increment`.
  - `DELETE` se você estiver apagando registros de uma tabela que tem chaves estrangeiras com a propriedade `ON DELETE CASCADE`.
  - `REPLACE ... SELECT`, `INSERT IGNORE ... SELECT` se você tiver valores de chaves duplicados nos dados inseridos.

  Se e somente se todos estas consultas NÃO tiverem cláusulas `ORDER BY` garantindo uma ordem determinística.

  Na verdade, por exemplo para `INSERT ... SELECT` sem `ORDER BY`, o `SELECT` pode retornar registros em uma ordem diferente (no qual resultará em um registro tendo diferentes posições, obtendo um número diferente na coluna `auto_increment`), dependendo da escolhe feita pelo otimizador no master e slave. Uma consulta será otimizada deiferentemente no master e slave apenas se:
  - Os arquivos usados pelas duas consultas não são exatamente a mesma; por exemplo `OPTIMIZE TABLE` foi executado nas tabelas master e não nas nas tabelas slave (para corrigir isto, desde o MySQL 4.1.1, `OPTIMIZE`, `ANALYZE` e `REPAIR` são escritos no log binário).
  - A tabela está armazenada em um mecanismo de armazenamento diferente no master e no slave (pode se executar diferentes mecanismos de armazenamento no metre e no slave: por exemplo, InnoDB ne master e MyISAM no slave, se o slave possuir menos espaço dispponível em disco).
  - The MySQL buffers' sizes (`key_buffer_size` etc) are different on the master and slave.
  - O master e slave executam versões diferentes do MySQL, e o código do toimizador é diferente entre estas versões.

Este problema também pode afetar a restauração de um banco de dados usando `mysqlbinlog|mysql`.

O modo mais fácil de evitar este problema em todos os casos é adicionar uma cláusula `ORDER BY` para tal consulta não determinística assegure que os registros são sempre armazenados/modificados na mesma ordem. Nas versões futuras do MySQL adicionaremos automaticamente uma cláusula `ORDER BY` quando necessário.

Os seguintes problemas são conhecidos e serão corrigidos na hora certa:

- Ao usar funções `RPAD`, ou qualquer outra função string que termina adicionando espaços em branco a direita, em uma consulta que preisa usar tabelas temporárias para ser rsolvida, todas as strings resultantes serão cortadas a direita (como em RTRIM). Este é um exemplo de uma consulta:

  ```
  SELECT RPAD(t1.field1, 50, ' ') AS f2, RPAD(t2.field2, 50, ' ') AS f1
  FROM table1 as t1 LEFT JOIN table2 AS t2 ON t1.record=t2.joinID ORDER BY
  t2.record;
  ```

  O resultado final deste erro é que o usuário não conseguira espaços em branco do lado direito do campo resultante.

  O comportamento anterior existe em todas as versões do MySQL.

  A razão disto é devido ao fato de tabelas HEAP, que são usadas primeiro para tabelas temporárias, não são capazes de tratar colunas VARCHAR.

  Este comportamento será corrigido em uma das distribuições da série 4.1.

- Devido ao modo como os arquvos de definições de tabelas são armazenados não se pode usar 255 caracteres (`CHAR(255)`) em nomes de tabelas, nomes de colunas e enum. Isto está programado para ser corrigido na versão 5.1 quando temos novos arquivos de formatos de definição de tabelas.
- Quando estiver usando `SET CHARACTER SET`, não é permitido usar caracteres especias no nome do banco de dados, tabelas ou campos.
- Pode-se usar `_` ou `%` com `ESCAPE` em `LIKE ... ESCAPE`.
- se você tiver uma coluna `DECIMAL` com um número armazenado em diferentes formatos (+01.00, 1.00, 01.00), `GROUP BY` pode considerar cada valor como um valor diferente.
- `DELETE FROM merge_table` usado sem `WHERE` irá apenas apagar o mapeamento para a tabela, não apagando tudo nas tabelas mapeadas.
- Você não pode construir em outro diretório quando estiver utilizando MIT-pthreads. Como isto necessitaria de alterações na MIT-pthreads, nós não estamos aptos a corrigí-la.
- `BLOB` valores não podem ser usados com confiança em `GROUP BY`, `ORDER BY` ou `DISTINCT`. Somente os primeiros bytes (padrão 1024) `max_sort_length` são usados quando estiver comparando `BLOB`s nestes casos. Isto pode ser alterado com a opção `-0 max_sort_lenght` para `mysqld`. Uma forma de contornar este problema para a maioria dos casos é usar a substring: `SELECT DISTINCT LEFT(blob,2048) FROM nome_tabela`.
- Cálculos são feitos com `BIGINT` ou `DOUBLE` (normalmente, ambos tem o tamanho de 64 bits). Depende da precisão utilizada na função. A regra geral é que funções binárias são feitas com precisão `BIGINT`, `IF` e `ELT()` com precisão `BIGINT` ou `DOUBLE` e o resto

com precisão DOUBLE. Devemos evitar o uso de valores sem sinal maiores que 63 bits (9223372036854775807) para qualquer outra coisa além de campos binários!

- Todas os campos string, exceto campos do tipo BLOB e TEXTO tem, automaticamente, todos os espaços extras removidos quando recuperados. Para tipos CHAR, isto não tem problema, e pode ser considerado como um recurso de acordo com o ANSI SQL92. O problema é que no MySQL, campos VARCHAR são tratados desta mesma forma.
- Você só pode ter até 255 colunas ENUM e SET em uma tabela.
- Em MIN(), MAX() e outras funções de agrupamente, o MySQL atualmente compara as colunas ENUM e SET pelo valor de suas strings ao invés da posição relativa da string no conjunto.
- mysqld_safe redireciona todas as mensagens de mysqld para o log mysqld. Um problema com isto é que se você executar o mysqladmin refresh para fechar e reabrir o log, a stdout e a stderr continuam redirecionadas para o log antigo. Se você utiliza --log extensivamente, deverá editar o mysqld_safe para logar em ‘'hostname'.err’ em vez de ‘'hostname'.log’; assim você pode facilmente utilizar o espaço do log antigo apagando-o e executando mysqladmin refresh.
- Em instruções UPDATE, colunas são atualizadas da esquerda para a direita. Se você referenciar a uma coluna atualizada, você irá obter o valor atualizado em vez do valor original, por exemplo:

  ```
  mysql> UPDATE nome_tabela SET KEY=KEY+1,KEY=KEY+1;
  ```

  Isto atualiza KEY com 2 no lugar de 1.
- Você pode se referir a múltiplas tabelas em uma mesma consulta, mas você não pode se referir a qualquer tabelas temporárias dada mais de uma vez. Por exemplo, a seguinte instrução não funciona.

  ```
  mysql> SELECT * FROM temporary_table, temporary_table AS t2;
  ```
- RENAME não funciona com tabelas temporárias (TEMPORARY) ou tabelas usadas em uma tabelas MERGE.
- O otimizador pode lidar com o DISTINCT de forma diferente se você estiver usando colunas 'escondidas' em uma join ou não. Em uma join, colunas escondidas são contadas como parte do resultado (mesmo se elas não são mostradas) enquanto que em queries normais colunas escondidas não participam na comparação DISTINCT. Nós provavelmente iremos alterar isto no futuro para nunca comparar as colunas escondidas quando executando DISTINCT.

  um exemplo disto é:

  ```
  SELECT DISTINCT mp3id FROM band_downloads
         WHERE userid = 9 ORDER BY id DESC;
  ```

  and

  ```
  SELECT DISTINCT band_downloads.mp3id
         FROM band_downloads,band_mp3
         WHERE band_downloads.userid = 9
         AND band_mp3.id = band_downloads.mp3id
         ORDER BY band_downloads.id DESC;
  ```

  No segundo caso, você pode obter duas linhas idênticas no MySQL 3.23.x na série do resultado (porque o campo escondido 'id' pode variar).

Perceba que isto somente acontece em consultas onde você não tem colunas ORDER BY no resultado, algo não permitido no SQL-92.

- Como o MySQL permite trabalhar com tipos de tabelas que não suportam transações (e assim não pode fazer `rollback` em dados) algumas coisas funcionam um pouco diferentes de outros servidores SQL em MySQL (Isto serve para garantir que o MySQL nunca necessitará de um rollback para um comando SQL). Porém isto pode ser um pouco estranho em casos que os valores dos campos devem ser verificados na aplicação, mas isto ira fornacer um ótimo ganho de velocidade assim como permite ao MySQL fazer algumas otimizações que de outro modo seriam muito difíceis para serem feitas.

  Se você informar um valor incorreto em uma coluna, o MySQL, em vez de fazer um rollback, aramzenará o `melhor valor possível` no campo.

  - Se tentar armazenar um valor fora da faixa em uma coluna numérico, o MySQL irá armazenar o menor ou maior valor possível no campo.
  - Se tentar armazenar uma string que não comece com um número em uma coluna numérica, o MySQL irá armazenar 0 na coluna.
  - Se você tentar armazenar `NULL` em uma coluna que não aceita valores nulos, MySQL irá armazenar 0 ou '' (string vazia) na coluna. (Este comportamento pode, entretanto, ser alterado com a opção de compilação -DDONT_USE_DEFAULT_FIELDS).
  - O MySQL permite o armazenamento de alguns valores errados de data em campos do tipo `DATE` e `DATETIME`. (Como 2000-02-31 ou 2000-02-00). A idéia é que não é serviço do servidor SQL validar datas. Se o MySQL pode armazenar uma data e recuperar extamente a mesma data, então o MySQL armazenará a data. Se a data estiver totalmente errada, o MySQL irá armazenar a data 0000-00-00 no campo.
  - Se você especificar um valor não suportado para um campo do tipo `enum`, ele será alterado para o valor de erro 'empty string', com valor numérico 0.
  - Se você definir uma coluna `SET` com um valor não suportado, o valor será ignorado.
- Se você executar uma `PROCEDURE` em uma pesquisa que retorna uma série vazia, em alguns casos a instrução `PROCEDURE` não irá transformar as colunas.
- Criação da tabela do tipo `MERGE` não verifiva se as tabelas envolvidas são de tipos compatíveis.
- O MySQL ainda não pode lidar com valores `NaN`, `-Inf` e `Inf` em tipos double. Usá-los causará problemas na exportação e importação de dados. Uma solução intermediária é alterar `NaN` para `NULL` (se for possível) e `-Inf` e `Inf` para o valor `double` mínimo ou máximo respectivo possível.
- Se você usar `ALTER TABLE` para primeiro adicionar um índice `UNIQUE` a uma tabela usada em uma tabela `MERGE` e então usar `ALTER TABLE` para adicionar um índice normal na tabela `MERGE`, a ordem das chaves será diferente para as tabelas se existir uma chave antiga não única na tabela. Isto é porque o `ALTER TABLE` coloca chaves `UNIQUE` antes de chaves normais para ser possível detectar chaves duplicadas o mais cedo o possível.

Os seguintes erros são conhecidos em versões mais antigas do MySQL:

- Você pode pendurar um processo se você fizer um `DROP TABLE` em uma tabela entre outras que esteja travada com `LOCK TABLES`.

- No caso seguinte você pode obter um descarrego de memória para o arquivo core:
  - Tratamento de inserções com atraso tem deixado inserções pendentes na tabela.
  - `LOCK table` com `WRITE`
  - `FLUSH TABLES`
- Antes da versão 3.23.2 do MySQL um `UPDATE` que atualizava uma chave com um `WHERE` na mesma chave podia falhar porque a chave era usada para procurar por registros e a mesma linha poderia ter encontrado vários itens:

  ```
  UPDATE nome_tabela SET KEY=KEY+1 WHERE KEY > 100;
  ```

  Um modo de contornar este erro é utilizar:

  ```
  mysql> UPDATE nome_tabela SET KEY=KEY+1 WHERE KEY+0 > 100;
  ```

  Isto funcionará porque MySQL não utilizará indices em expressões com a cláusula `WHERE`.
- Antes da versão 3.23 do MySQL, todos os tipos numéricos tratados como campos de pontos fixos. Isto significa que você tem que especificar quantas casas decimais um campo de ponto flutuante deve ter. Todos os resultados eram retornados com o número correto de casas decimais.

Para erros específicos na plataforma, vejas as seções sobre compilação e portabilidade. Veja Seção 2.3 [Installing source], Página 94. Veja Apêndice E [Porting], Página 1078.

2 Instalação do MySQL

Este capítulo descreve como obter e instalar o MySQL:

- Para uma lista de sites no quais você pode obter o MySQL, veja Seção 2.2.1 [Getting MySQL], Página 75.
- Para saber quais são as plataformas suportadas, veja em Seção 2.2.3 [Which OS], Página 78. Por favor perceba que nem todas as plataformas suportadas são igualmente boas para executar o MySQL. Algumas são mais robustas e eficientes que outras - ver Seção 2.2.3 [Which OS], Página 78 para detalhes.
- Várias versões do MySQL estão disponíveis em distribuições binárias e fonte. Nós também fornecemos acesso público à nossa árvore fonte atual para aqueles que desejam ver nossos desenvolvimentos mais recentes e nos ajudar a testar novos códigos. Para determinar que versão e tipo da distribuição você deve usar, veja Seção 2.2.4 [Which version], Página 80. Se ainda restar dúvidas, use uma a distribuição binária.
- Instruções de instalação para distribuições binária e fonte são descritos em Seção 2.2.9 [Installing binary], Página 91 e Seção 2.3 [Installing source], Página 94. Cada conjunto de instruções inclui uma seção sobre problemas específicos de sistemas que você pode precisar.
- Para procedimentos pós-instalação, veja Seção 2.4 [Post-installation], Página 111. Estes procedimentos podem ser aplicados caso você use uma distribuição binária ou fonte do MySQL.

2.1 Instalação rápida padrão do MySQL

Este capítulo cobre a instalação do MySQL em plataformas onde oferecemos pacotes usando oformato de empacotamento nativo da respectiva plataforma. No entanto, as distribuições binárias do MySQL estão disponíveis para muitas outras plataformas, veja Seção 2.2.9 [Installing binary], Página 91 para instruções gerais de instalação para estes pacotes que se aplicam a todas as plataformas.

Veja Seção 2.2 [General Installation Issues], Página 75 para mais informações sobre quais outras distribuições binárias estão disponíveis e como obtê-las.

2.1.1 Instalando o MySQL no Windows

O processo de instalação para o MySQL no Windows tem os seguintes passos:

1. Instale a distribuição.
2. Configure um arquivo de opção se necessário.
3. Selcione o servidor que você quer usar.
4. Inicie o servidor.

O MySQL para Windows está disponível em dois formatos de distribuição:

- A distribuição binária contém um programa de instalação que instala que você precisa e assim possa iniciar o servidor imediatamente.

- A distribuição fonte contém todo o código e os arquivos suportados para construir o executável usando o compilador VC++ 6.0. Veja Seção 2.3.7 [Construção do fonte para Windows], Página 107.

Geralmente, a melhor opção é a distribuição binária. É mais simples e você não precisa de nenhuma ferramenta adicional para ter o MySQL em execução.

2.1.1.1 Exigências do Sistema Windows

Para executar o MySQL no Windows, você precisará do seguinte:

- Um sistema operacional Windows de 32 bits como 9x, ME, NT, 2000 ou XP. A famílía NT (Windows NT, 2000 e XP) lhe permite executar o servidor MySQL como um serviço. Veja Seção 2.1.1.7 [NT start], Página 66.
- Suporte ao protocolo TCP/IP.
- Um cópia da distribuição binária do MySQL para Windows, o qual pode ser feito download em `http://www.mysql.com/downloads/`.

  Nota: A distribuição de arquivos são fornecidas no formato zip e recomendamos o uso de um cliente FTP com opção de resumo para evitar corrompimento de arquivos durante o processo de download.
- Um programa `ZIP` para descompactar os arquivos da distribuição.
- Espaço suficiente em disco para descompactar, instalar e criar o banco de dados de acordo com suas exigências.
- Se você planeja se conectar ao servidor MySQL via ODBC, você também precisará do dirver `MyODBC`. Veja Seção 12.2 [ODBC], Página 870.
- Se você precisa de tabelas com tamanho maior que 4GB, instale o MySQL em um sistema de arquivos NTFS ou mais novo. Não se esqueça de usar `MAX_ROWS` e `AVG_ROW_LENGTH` quando criar tabelas. Veja Seção 6.5.3 [`CREATE TABLE`], Página 598.

2.1.1.2 Instalando uma Distribuição Binária do Windows

Para instalar o MySQL no Windows usando uma distribuição binária, siga este procedimento:

1. Se você estiver trabalhando em uma máquina Windows NT, 2000, ou XP, esteja certo de que você está logado com um usuário com privileios de administrador.
2. Se você estiver fazendo uma atualização de uma instalação MySQL mais nova, é necessário parar o servidor atual. Em máquinas com Windows NT, 200 ou XP, se você estiver executando o servidor como um serviço, pare-o com o comando:

   ```
   C:\> NET STOP MySQL
   ```

   Se você planeja usar um servidor diferente depois da atualização (por exemplo, se você quiser executar o `mysqld-max` em vez do `mysqld`), remova o serviço existente:

   ```
   C:\mysql\bin> mysqld --remove
   ```
3.

   Você pode reinstalar o serviço com o servidor próprio depois de atualizar.

   Se você não estiver executando o servidor MySQL como um serviço, pare desta forma:

```
C:\mysql\bin> mysqladmin -u root shutdown
```

4. Finalize o programa `WinMySQLAdmin` se ele estiver em execução.
5. Descompacte os arquivos de distribuição em um diretório temporário.
6. Execute o programa '`setup.exe`' para iniciar o processo de instalação. Se você quiser instalar em um diretório diferente do padrão ('`c:\mysql`'), use o botão `Browse` para especificar seu diretório preferido. Se você não instalar o MySQL no local padrão, você precisará epecificar o local onde você inicia o servidor. O modo mais fácil de se fazer isto é usar um arquivo de opção, como descrito em Seção 2.1.1.3 [Windows prepare environment], Página 62.
7. Finalize o processo de instalação.

2.1.1.3 Preparando o Ambiente MySQL do Windows

Se você precisar especificar opções de inicialização quando executar o servidor, você pode indentifica-los na linha de comando ou colocá-los em um arquivo de opção. Par opções que são usadas sempre que o servidor iniciar, você achará mais conveniente utilizar um arquivo de opcão para especificar a configuração do seu MySQL. Isto é particularmente verdade sob as seguintes circunstâncias:

- A localização do diretório de instalação ou dados são diferentes dos locais padrão ('`c:\mysql`' e '`c:\mysql\data`').
- Você precisa ajustar as configurações do servidor. Por exemplo, para usar as tabelas transacionais `InnoDB` no MySQL versão 3.23, você deve criar manualmente dois novos diretórios para guardar os arquivos de dados e de log do `InnoDB` — por exemplo, '`c:\ibdata`' e '`c:\iblogs`'. Você também poderá adicionar algumas linhas extras ao arquivo de opção, como descrito em Seção 7.5.3 [Iniciando o `InnoDB`], Página 644. (A partir do MySQL 4.0, o InnoDB cria os seus arquivos de log e dados no diretório de dados por padrão. Isto significa que você não precisa configurar o InnoDB explicitamente. Você ainda deve fazê-lo se desejar, e um arquivo de opção será útil neste caso.)

No Windows, o instalador do MySQL coloca o diretório de dados diretamente sob o diretório onde você instalou o MySQL. Se você quisesse utilizar um diretório de dados em um local diferente, você deve copiar todo o conteúdo do diretórios `data` para a nova localização. Por exemplo, por padrão, o instalador coloca o MySQL em '`C:\mysql`' e o diretório de dados em '`C:\mysql\data`'. Se você quiser usar um diretório de dados de '`E:\mydata`', você deve fazer duas coisas:

- Mova o diretório de dados de '`C:\mysql\data`' para '`E:\mydata`'.
- Use uma opção `--datadir` para especificar a nova localização do diretório de dados cada vez que você iniciar o servidor.

Quando o servidor MySQL inicia no Windows, ele procura pelas opções em dois arquivos: O arquivo '`my.ini`' no diretório Windows e o arquivo chamado '`C:\my.cnf`'. O diretório do Windows é normalmente chamado '`C:\WINDOWS`' ou '`C:\WinNT`'. Você pode determinar a sua localização exata a partir do valor da variável de ambiente `WINDIR` usando o seguinte comando:

```
C:\> echo %WINDIR%
```

O MySQL procura pelas opções primeiro no arquivo '`my.ini`', e então pelo arquivo '`my.cnf`'. No entanto, para evitar confusão, é melhor se você usar apenas um destes arquivos. Se o seu PC usa um boot loader onde o drive `C:` não é o drive de boot, sua única opção é usar o arquivo '`my.ini`'. Independente de qual arquivo usar, ele deve ser no formato texto.

Um arquivo de opção pode ser criado e modificado com qualquer editor de texto como o programa `Notepad`. Por exemplo, se o MySQL está instalado em '`D:\mysql`' e o diretório de dados está localizado em '`D:\mydata\data`', você pode criar o arquivo de opção e definir uma seção `[mysqld]` para especificar valores para os parâmetros `basedir` e `datadir`:

```
[mysqld]
# defina basedir com o seu caminho de instalação
basedir=D:/mysql
# defina datadir com o local do diretório de dados,
datadir=D:/mydata/data
```

Note que os nome de caminho do Windows são especíificados em arquivos de opção usando barras normais em ves de barra invertida. Se você usar barras invertidas, você deve usá-las em dobro.

Outro modo de se gerenciar um arquivo de opção é usar a ferramenta `WinMySQLAdmin`. Você pode encontrar o `WinMySQLAdmin` no diretório '`bin`' de sua instalação MySQL, assim como um arquivo de ajuda contendo instruções para usá-lo. O `WinMySQLAdmin` tem a capacidade de editar os seus arquivos de opção, mas note o seguinte:

- `WinMySQLAdmin` usa apenas o arquivo '`my.ini`'.
- Se o `WinMySQLAdmin` encontra o arquivo '`C:\my.cnf`', ele o renomeará para '`C:\my_cnf.bak`' para disabilitá-lo.

Agora você está pronto para testar o servidor.

2.1.1.4 Selecionando um Servidor Windows

Iniciado com o MySQL 3.23.38, a distribuição Windows inclui ambos binários, normal e o MySQL-Max. Aqui está uma lista dos diferentes servidores MySQL dos quais você pode escolher:

Binario	Descrição
`mysqld`	Compilado com debugger integral e conferência automática de alocação de memória, links simbólicos, `BDB` e tabelas `InnoDB`.
`mysqld-opt`	Binário otimizado. A partir da versão 4.0 o `InnoDB` está habilitado. Antes desta versão, este servidor não tem suporte a tabelas transacionais.
`mysqld-nt`	Binário otimizado para NT/2000/XP com suporte para named pipes.
`mysqld-max`	Binário otimizado com suporte para links simbólicos, tabelas `BDB` e `InnoDB`.
`mysqld-max-nt`	Como o `mysqld-max`, porém compilado com suporte para named pipes.

Todos os binários acima são otimizados para processadores Intel modernos mas deve funcionar em qualquer processador Intel i386 ou melhor.

Os servidores `mysqld-nt` e `mysqld-max-nt` suportam conexões named pipe. Se você usar um destes servidores, o uso de named pipes está sujeito a estas condições:

- Os servidores devem ser executados em uma versão do Windows que suporte named pipes (NT, 2000, XP).
- A partir da versão 3.23.50, named pipes só estarão habilitados se você iniciar estes servidores com a opção `--enable-named-pipe`.
- Os servidores podem ser executados no Windows 98 ou Me, mas o TCP/IP deve estar instalado, e as conexões named pipes não podem ser usadas.
- No Windows 95, estes servidores não podem ser usados.

2.1.1.5 Iniciando o Servidor pela Primeira Vez

No Windows 95, 98, ou Me, cliente MySQL sempre se conecta ao servidor usando TCP/IP. Nos sistemas baseados no NT, como o Windows NT, 2000, ou XP, os clientes possuem duas opções. Eles podem usar TCP/IP, ou eles podem usar um named pipe se o servidor suportar conexões named pipes.

Para informações sobre qual servidor binário executar, veja Seção 2.1.1.3 [Windows prepare environment], Página 62.

Esta seção lhe dá um visão geral da inicialização de um servidor MySQL. A seguinte seção fornce informação mais específica para versões particulares do Windows.

Os exemplos nesta seção assumem que o MySQL está instalado sob a localização padrão, '`C:\mysql`'. Ajuste o caminho mostrado nos exemplos se você tiver o MySQL instalado em um local diferente.

Fazer um teste a partir do prompt de comando do em uma janela de console (uma janela "DOS") é a melhor coisa a fazer porque o servidor mostra a mensagem de status que aparece na janela do DOS. Se alguma coisa estiver errado com sua configuração, estas mensagens tornarão mais fácil para você de identificar e corrigir qualquer problema.

Tenha certeza que você está no diretório onde o servidor é localizado e então entre este comando:

```
shell> mysqld --console
```

Para servidores que incluem suporte `InnoDB`, você deve ver as seguintes mensagens assim que o servidor iniciar:

```
InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file c:\ibdata\ibdata1 size to 209715200
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
```

```
011024 10:58:25  InnoDB: Started
```

Quando o servidor finaliza sua sequência de inicialização, você deve ver algo como abaixo, que indica que o servidor está pronto para o conexão com o cliente:

```
mysqld: ready for connections
Version: '4.0.14-log'  socket: ''  port: 3306
```

O servidor continuará a gravar no console qualquer saída de diagnóstico adicional que ele produza. Você pode abrir uma nova janela de console na qual se executará os programas clientes.

Se você omitir a opção `--console`, o servidor grava a saída do diagnóstico no log de erro no diretório de dados. O log de erro é o arquivo com a extensão '`.err`'.

2.1.1.6 Iniciando o MySQL no Windows 95, 98, ou Me

No Windows 95, 98 ou Me, o MySQL usa TCP/IP para conectar um cliente a um servidor. (Isto permitirá que qualquer máquina na sua rede se conecte a seu servidor MySQL.) Por isto, você deve ter certeza de que o suporte TCP/IP está instalado na sua máquina antes de iniciar o MySQL. Você pode encontrar o TCP/IP no seu CD-ROM do Windows.

Note que se você estiver usando uma versão antiga do Win95 (por exemplo, OSR2). É preferível que você use um pacote antigo Winsock; para o MySQL é necessário o Winsock 2! Você pode obter o Winsock mais novo em `http://www.microsoft.com`. O Windows 98 tem a nova biblioteca Winsock 2, portanto não é necessário atualizar a biblioteca.

Para iniciar o servidor `mysqld`, você deve iniciar uma janela do Prompt (Janela "MS-DOS") e digitar:

```
shell> C:\mysql\bin\mysqld
```

Isto irá iniciar o `mysqld` em segundo plano. Isto é, depois do servidor iniciar, você deve ver outro prompt de comando. (Note que se você iniciar o servidor deste modo no Windows NT, 2000 ou XP, o servidor irá executar em segundo plano e nenhum prompt de comando aparecerá até que o servidor finalize. Por isto, você deve abrir outro prompt de comando para executar programas clientes enquanto o servidor estriver em execução.)

Você pode finalizar o servidor MySQL executando:

```
shell> C:\mysql\bin\mysqladmin -u root shutdown
```

Isto chama o utilitário administrativo do MySQL `mysqladmin` para conectar ao servidor e manda-lo finalizar. O comando conecta como `root` que é a conta administrativa padrão no sistema de permissões do MySQL. Por favor, note que o sistema de permissões do MySQL é totalmente independente de qualquer login de usuário sob o Windows.

Se o `mysqld` não iniciar, por favor, verifique o log de erro para ver se o servidor escreveu alguma mensagem que possa indicar a causa do problema. Você pode também tentar iniciar o servidor com `mysqld --console`; neste caso, você pode obter alguma informação útil na tela que pode ajudar a resolver o problema.

A última opção é iniciar o `mysqld` com `--standalone --debug`. Neste caso o `mysqld` irá escrever em um arquivo log '`C:\mysqld.trace`' que deve conter a razão pela qual o `mysqld` não inicia. Veja Seção E.1.2 [Making trace files], Página 1080.

Use `mysqld --help` para mostrar todas as opções que o `mysqld` entende!

2.1.1.7 Iniciando o MySQL no Windows NT, 2000, ou XP

Na famílía NT (Windows NT, 2000 ou XP) o modo recomendado de executar o MySQL é instalá-lo como um serviço do Windows. O Windows então inicia e para o servidor MySQL automaticamente quando o Windows inicia e para. Um servidor instalado como um serviço também pode ser controlado a partir da linha de comando usando os comandos `NET`, ou com o utilitário gráfico `Serviços`.

O utilitário `Serviços` (o `Service Control Manager` do Windows) pode ser encontrado no `Painel de Controle` do Windows (em `Ferramentas Administrativas` no Windows 2000). É recomendado que se feche o utilitário `Serviços` enquanto realiza a operações de instalação ou remoção do servidor a partir desta linha de comando. Isto evita alguns erros estranhos.

Para ter o MySQL funcionando com TCP/IP no Windows NT 4, você deve instalar o service pack 3 (ou mais novo)!

Antes de instalar o MySQL como um serviço, você deve primeiro parar o servidor atual em execução usando o seguinte commando:

```
shell> C:\mysql\bin\mysqladmin -u root shutdown
```

Isto chama o utilitário administrativo do MySQL `mysqladmin` para conectar ao servidor e mandá-lo parar. O comando conecta com `root` que é a conta administrativa padrão no sistema de permissões do MySQL. Por favor, note que o sistema de permissões do MySQL é totalmente independente de qualquer login de usuário sob o Windows.

Agora instale o servidor como um serviço:

```
shell> mysqld --install
```

Se você não definir um nome para o serviço, ele é instalado com o nome `MySQL`. Uma vez instalado, ele pode ser imediatamente iniciado a partir do utilitário `Serviços`, ou usando o comando `NET START MySQL`. (Este comando é caso insensitivo).

Uma vez em execução, o `mysqld` pode ser parado usando o utilitário de Serviços ou usando o comando `NET STOP MySQL`, ou o comando `mysqladmin shutdown`.

Se você tiver problemas instalando o `mysqld` como um servico usando apenas o nome do servidor, tente instalá-lo usando seu caminho compelto:

```
shell> C:\mysql\bin\mysqld --install
```

A partir do MySQL 4.0.2, você pode especificaro nome do serviço depois da opção `--install`. A partir do MySQL 4.0.3, você pode especificar uma opção `--defaults-file` depois do nome do serviço para indicar onde o servidor deve obter opções ao iniciar. A regras que determinam o nome do serviço e os arquivos de opção que o servidor usa são as seguintes:

- Se você não especificar um nome de serviço, o servidor usa o nome padrão do `MySQL` e o servidor lê as opções do grupo `[mysqld]` no arquivo de opções padrão.
- Se você especificar um nome de serviço depois da opção `--install`, o servidor ignora o grupo de opção `[mysqld]` em vez de ler opções do grupo que tem o mesmo nome que o serviço. O servidor le opções do arquivo de opções padrão.
- Se você especificar uma opção `--defaults-file` depois do nome de serviço, o servidor ignora o arquivo de opções padrão e lê opções apenas a partir do grupo `[mysqld]` do arquivo indicado.

Nota: Antes do MySQL 4.0.17, um servidor instalado como um serviço do Windows tinha problema na inicialização se o seu caminho ou nome do serviço possuisse espaços. Por esta razão, evite instalar o MySQL em um diretório como '`C:\Program Files`' ou usar um nome de serviço contendo espaço.

No caso normal que você instala o servidor com `--install` mas nenhum nome de serviço, o servidor é instalado com um nome de serviço de `MySQL`.

Como um exemplo mais complexo, considere o seguinte comando:

```
shell> C:\mysql\bin\mysqld --install mysql --defaults-file=C:\my-opts.cnf
```

Aqui, um nome de serviço é dado depois de opção `--install`. Se nenhuma opção `--defaults-file` for dada, este comando teria o efeito de fazer o servidor ler o grupo `[mysql]` a partir do arquivo de opções padrão. (Isto seria uma má idéia, porque aquele grupoo de opção é para ser usado pelo programa cliente `mysql`.) No entanto, como a opção `--defaults-file` está presente, o servidor lê as opções apenas a partir do arquivo indicado, e apenas do grupo de opção `[mysqld]`.

Você também pode especificar as opções como "`Parâmetros de inicialização`" no utilitário de `Serviços` do Windows antes de você iniciar o serviço MySQL.

Uma vez que o servidor MySQL é instalado, o Windows irá iniciar o serviço automaticamente sempre que o Windows inicia. O serviço também pode ser iniciado imediatamente a partir do utilitário `Serviços` ou usando o comando `NET START MYSQL`. O comando `NET` não é caso sensitivo.

Note que quando executado como um serviço, o `mysqld` não têm acesso a um console e então nenhuma mensagem pode ser vista. Se o `mysqld` não iniciar, verifique o log de erros par ver se o servidor gravou alguma mensagem lá indicando a causa do problema. O log de erro está localizado no diretório '`c:\mysql\data`'. É o arquivo com um sufixo '`.err`'.

Quando o `mysqld` está executando como um serviço, ele pode ser parado usando o utilitários `Serviços`, o comando `NET STOP MYSQL`, ou o comando `mysqladmin shutdown`. Se o serviçp estiver em execução quando o Windows desliga, o Windows irá parar o servidor automaticamente.

A partir do MySQL versão 3.23.44, você pode escolher entre instalar o servidor como um serviço `Manual` se você não deseja que os serviços sejam executados automaticamente durante o processo de inicialização. Para fazer isto, use a opção `--install-manual` em vez da opção `--install`.

```
shell> C:\mysql\bin\mysqld --install-manual
```

Para remover um serviço que está instalado como um serviço, primeiro pare-o se ele estiver em execução. Então use a opção `--remove` para removê-lo:

```
shell> mysqld --remove
```

Um problema com a finalização automática do serviço MySQL é que, para versões do MySQL anteriores a 3.23.49, o Windows esparava apenas por alguns segundos para o desligamento completo, e matava os processos do servidor de banco de dados se o tempo limite fosse excedido. Isto potencialmente causava problemas. (Por exemplo, o mecanimo de armazenamento `InnoDB` deverá fazer uma recuperação de falhas na próxima inicialização). A partir do MySQL 3.23.49, o Windows irá esperar mais para que a finalização do MySQL Server esteja completa. Se você notar que ainda não é o suficiente para a sua instalação,

não é seguro executar o MySQL Server como um serviço. Em vez disso, execute-o a partir do prompt de comando, e finalize-o com `mysqladmin shutdown`.

A alteração para avisar para o Windows para esperar mais quando parar o servidor MySQL funciona apenas com o Windows 2000 e XP, mas não para o Windows NT. No NT, o Windows espera apenas 20 segundos para que o serviço seja finalizado, e depois desso ele mata o processo do serviço. Você pode aumentar este padrão abrindo o Editor de Registro (‘`\winnt\system32\regedt32.exe`’) e editar o valor de `WaitToKillServiceTimeout` em ‘`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control`’ na árvore do Registro. Especifique o novo valor mais largo em milisegundos (por exemplo 12000 para que o Windows NT espere até 120 segundos).

Se você não quiser iniciar o `mysqld` como um serviço, você pode iniciá-lo a partir da linha de comando do mesmo modo que em versões do Windows que não são baseados no NT. Para instruções use Seção 2.1.1.6 [Win95 start], Página 65.

2.1.1.8 Executando o MySQL no Windows

O MySQL suporta TCP/IP em todas as plataformas Windows. Os servidores `mysqld-nt` e `mysql-max-nt` suportam named pipes no NT, 2000 e XP. No entanto, o padrão é usar TCP/IP, independente da plataforma:

- Named pipes é atualmente mais lento que TCP/IP em muitas configurações do Windows.
- Alguns usuários encontraram problemas ao finalizar o servidor MySQL quando era usado named pipes.

A partir da versão 3.23.50, named pipes só está habilitado para o `mysqld-nt` e `mysql-max-nt` se eles forem iniciados com a opção `--enable-named-pipe`.

Você pode forçar que um cliente MySQL use named pipes especificando a opção `--pipe` ou especificando `.` como nome de máquina. Use a opção `--socket` para especificar o nome do pipe. No MySQL 4.1, você deve usar a opção `--protocol=PIPE`.

Você pode testar se o MySQL está funcionando executando qualquer dos seguintes comandos:

```
C:\> C:\mysql\bin\mysqlshow
C:\> C:\mysql\bin\mysqlshow -u root mysql
C:\> C:\mysql\bin\mysqladmin version status proc
C:\> C:\mysql\bin\mysql test
```

Se o `mysqld` está lento para responder a suas conexões no Win95/Win98, provavelmente existe um problema com seu DNS. Neste caso, inicie o `mysqld` com a opção `--skip-name-resolve` e use somente `localhost` e números IP na coluna `Host` das tabelas de permissões do MySQL.

Existem duas versões da ferramenta de linha de comando MySQL:

Binario	Descrição
`mysql`	Compilado em Windows nativo, oferecendo capacidades de edição de texto muito limitadas.
`mysqlc`	Compilado com o compilador Cygnus GNU, que oferece edição `readline`.

Se você desejar usar o `mysqlc`, deve ter uma cópia da biblioteca '`cygwinb19.dll`' em algum lugar que o `mysqlc` possa encontrá-la. Se sua distribuição do MySQL não tiver esta biblioteca instalada no mesmo diretório que o `mysqlc` (o diretório `bin` sob o diretório base sa dua instalação do MySQL). Se sua distribuição não tem a biblioteca `cygwinb19.dll` no diretório '`bin`', olhe no diretório `lib` para encontrá-lo e copiá-lo para o seu diretório de sistema no Windows. ('`\Windows\system`' ou um lugar parecido).

Os privilégios padrões no Windows dão a todos usuários locais privilégios totais para todos os bancos de dados sem necessidade de especificar uma senha. Para deixar o MySQL mais seguro, você deve configurar uma senha para todos os usuário e remover a linha na tabela `mysql.user` que tem `Host='localhost'` e `User=''`.

Você também deve adicionar uma senha para o usuário `root`. O exemplo seguinte exemplo inicia removendo o usuário anônimo que tem todos os privilégios, e então configura uma senha para o usuário `root`:

```
C:\> C:\mysql\bin\mysql mysql
mysql> DELETE FROM user WHERE Host='localhost' AND User='';
mysql> FLUSH PRIVILEGES;
mysql> QUIT
C:\> C:\mysql\bin\mysqladmin -u root password your_password
```

Depois de configurar a senha, se você desejar desligar o servidor `mysqld`, você pode usar o seguinte comando:

```
C:\> mysqladmin --user=root --password=sua_senha shutdown
```

Se você estiver usando o servidor de uma antiga versão shareware do MySQL versão 3.21m o comando `mysqladmin` para configurar uma senha irá falhar com um erro: `parse error near 'SET password'`. A correção para este problema é atualizar para uma versão mais nova do MySQL.

Com as versões atuais do MySQL você pode facilmente adicionar novos usuários e alterar privilégios com os comandos `GRANT` e `REVOKE`. Veja Seção 4.4.1 [GRANT], Página 254.

2.1.2 Instalando o MySQL no Linux

O modo recomendado para instalar o MySQL no Linux é usando um arquivo RPM. Os RPMs do MySQL atualmente são construídos na versão 7.3 do sistema Suse Linux mas deve funcionar em outras versões de Linux que suportam `rpm` e usam `glibc`.

Se você tiver problemas com um arquivo RPM (por exemplo, se você receber o erro "`Sorry, the host 'xxxx' could not be looked up`"), veja Seção 2.6.2.1 [Binary notes-Linux], Página 141.

Na maioria dos casos, você só precisa instalar os pacotes `servidor MySQL` e o `cliente MySQL` para ter uma instalação funcional do MySQL. Os outros pacotes não são exigidos para uma instalação padrão. Se você quiser executar um servidor MySQL Max que tenha capacidades adicionais, você deve instalar o RPM `MySQL-Max` depois de instalar o RPM `MySQL-server`. Veja Seção 4.8.5 [mysqld-max], Página 343.

Se você tiver um dependência de falha ao tentar instalar os pacotes do MySQL 4.0 (ex.: "`error: removing these packages would break dependencies: libmysqlclient.so.10 is needed by ...`"), você também deve instalar o pacote `MySQL-shared-compat`,

o qual inclui ambas as bibliotecas para compatibilidade com versões anteriores (`libmysqlclient.so.12` para MySQL 4.0 e `libmysqlclient.so.10` para MySQL 3.23).

Muitas distribuições Linux ainda vêm com o MySQL 3.23 a elas normalmente ligam as aplicações dinamicamente para economizar espaço em disco. Se estas bibliotecas compartilhadas estão em pacotes separados (ex.; `MySQL-shared`), é suficiente simplesmente deixar estes pacotes instalados e apenas atualizar os pacotes do servidor e cliente MySQL (que são estaticamente ligados e não dependem de bibliotecas compartilhadas). Para distribuições que incluem as bibliotecas compartilhadas no mesmo pacote que o servidor MySQL (ex.: Red Hat Linux), você também pode instalar nosso RPM `MySQL-shares` 3.23 ou usar o pacote compatível com `MySQL-shared`.

Os seguintes pacotes RPM estão disponíveis:

- `MySQL-server-VERSION.i386.rpm`

  O servidor MySQL. Você ira precisar dele a não ser que você apenas queira se conectar a um servidor MySQL executando em outra máquina. Note que este pacote era chamado `MySQL-VERSION.i386.rpm` antes do MySQL 4.0.10.

- `MySQL-Max-VERSION.i386.rpm`

  O servidor MySQL Max. Este seridor tem capacidades adicionais que o servidor no ROM `MySQL-server` não tem. Você deve instalar o RPM `MySQL-server` primeiro, porque o RPM `MySQL-Max` depende dele.

- `MySQL-client-VERSION.i386.rpm`

  Os programas clientes padrões do MySQL. Provavelmente você sempre instalará este pacote.

- `MySQL-bench-VERSION.i386.rpm`

  Testes e comparativos de performances (benchmarks). Necessita do Perl e módulos do `BDB-mysql`.

- `MySQL-devel-VERSION.i386.rpm`

  As bibliotecas e arquivos include necessários se você precisa para compilar outros clientes MySQL, como nos módulos Perl.

- `MySQL-shared-VERSION.i386.rpm`

  Este pacote contém as bibliotecas compartilhadas (`libmysqlclient.so*`) que certas linguagens e aplicações nencessárias para carregar dinâmicamente e usar o MySQL.

- `MySQL-shared-compat-VERSION.i386.rpm`

  Este pacote inclui o biblioteca compartilhada para MySQL 3.23 e MySQL 4.0. Instale este pacote em vez do `MySQL-shared`, se você tiver aplicações instaladas que são dinâmicamente ligadas ao MySQL 3.23 mas você quer atualizar para o MySQL 4.0 sem quebrar as dependências da biblioteca. Este pacote esta disponível desde o MySQL 4.0.13.

- `MySQL-embedded-VERSION.i386.rpm`

  A biblioteca do servidor embutido MySQL (MySQL 4.0).

- `MySQL-VERSION.src.rpm`

  Este contém o código fonte para todos os pacotes acima. Ele também pode ser usado para tentar construir RPMs para outras arquiteturas (por exemplo, Alpha ou SPARC).

Para ver todos os arquivo em um pacote RPM, (por exemplo, um RPM `MySQL-server`), execute:

```
shell> rpm -qpl MySQL-server-VERSION.i386.rpm
```

Para realizar uma instalação mínima padrão, execute:

```
shell> rpm -i MySQL-server-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

Para instalar somente o pacote cliente, execute:

```
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

O RPM fornece um recurso para verificar a integridade e autenticidade dos pacotes antes de instalá-los. Se você quiser aprender mais sobre este recurso, veja Seção 2.2.2 [Verifying Package Integrity], Página 75.

O RPM coloca dados sob o '`/var/lib/mysql`'. O RPM também cria as entradas apropriadas em '`/etc/rc.d/`' para iniciar o servidor automaticamente na hora do boot. (Isto significa que se você realizou uma instalação anterior e fez alterações em seu script de inicialização, você pode desejar criar uma cópia do script para que você não perca ao instalar um RPM mais novo). Veja Seção 2.4.3 [Automatic start], Página 118 para mais informações sobre como o MySQL pode ser iniciado automaticamente na inicialização do sistema.

Se você quiser instalar o RPM do MySQL em uma distribuição Linux mais antiga que não suporte scripts de inicialização no '`/etc/init.d`' (diretamente ou via link simbólico), você deve criar um link simbólico que aponte para a localização onde o seu script de instalação está atualmente instalado. Por exemplo, se esta localização for '`/etc/rc.d/init.d`', use estes comandos antes de intalar o RPM para criar '`/etc/init.d`' como um link simbólico que aponte lá:

```
shell> cd /etc; ln -s rc.d/init.d .
```

No entanto, todas as distribuições de Linux atuais já devem suportar este novo layout de diretório que usa '`/etc/init.d`' já que ele é exigido para compatibilidade LBS (Linux Standard Base).

Se o arquivo RPM que você instalar inclui o `MySQL-server`, o daemon `mysqld` deve estar pronto e em execução após a instalação. Agora você já deve poder iniciar o MySQL. Veja Seção 2.4 [Pós Instalação], Página 111.

Se alguma coisa der errado, você encontrar maiores informações no capítulo de instalação. Veja Seção 2.2.9 [Instalado o binário], Página 91.

2.1.3 Instalando o MySQL no Mac OS X

A partir do MySQL 4.0.11, você pode instalar o MySQL no Mac OS X 10.2 ("Jaguar") usando um pacote do binário do Mac OS X `PKG` em vez da distribuição binário em tarball. Note que versões mais antigas do Mac OS X (ex.: 10.1.x) não são suportadas por este pacote.

Este pacote está localizado dentro de um arquivo de imagem de disco (`.dmg`). que você primeiro precisa montar com um duplo clique em sua ícone no Finder. Ele deve então montar a imagem e exibir o seu conteúdo.

NOTA: Antes de proceder com a instalação, tenha certeza que você finalizou todas as instâncias do MySQL em execução usando o MySQL Manager Aplication (no Mac OS X Server) ou via `mysqladmin shutdown` na linha de comando.

Para relamente instalar o MySQL PKG, de um duplo clique na ícone do pacote. Isto inicia o Mac OS Package Installer, que irá guia-lo pela instalação do MySQL.

O Mac OS X PKG do MySQL irá se instalar em '`/usr/local/mysql-<version>`' e também instalrá um link simbólico '`/usr/local/mysql`', apontando para a nova localização. Se um diretório chamado '`/usr/local/mysql`' já existe, ele será renomeado para '`/usr/local/mysql.bak`' em primeiro lugar. Adicionalmente, ele irá instalar a tabela de permissões do banco de dados MySQL executando `mysql_install_db` depois da instalação.

O layout de instalação é similar a aquele da distribuição binária, todos os binários do MySQL estão localizados no diretório '`/usr/local/mysql/bin`'. O socket MySQL será colocado em '`/tmp/mysql.sock`' por padrão. Veja Seção 2.2.5 [Installation layouts], Página 83.

A instalação do MySQL exige uma conta de usuário do Mac OS X chamada `mysql` (uma conta de usuário com este nome existe por padrão no Mac OS X 10.2 e acima).

Se você estiver executando o MAC OS X Server, você já terá uma versão do MySQL instalado:

- Mac OS X Server 10.2-10.2.2 vem com o MySQL 3.23.51 instalado
- Mac OS X Server 10.2.3-10.2.6 vem com o MySQL 3.23.53
- Mac OS X Server 10.3 vem com o MySQL 4.0.14

Esta seção do manual cobre a instalação apenas do MySQL Mac OS X PKG oficial. Leia o ajuda da Apple sobre a instalação do MySQL (Execute o aplicativo "Help View", selecione a ajuda do "Mac OS X Server" e faça uma busca por "MySQL" e leia o item entitulado "Installing MySQL").

Note especialmente, que a versão pré-instalada do MySQL no Mac OS X Server é iniciado com o comando `safe_mysqld` em vez de `mysqld_safe`.

Se anteriormente você usava pacotes do MySQL de Marc Liyanage para Mac OS X de `http://www.entropy.ch`, você pode simplesmente seguir as intruções de atualização para pacotes usando o layout de instalação dos binário como dados em suas páginas.

Se você está atualizado da versão 3.23.xx de Marc ou do versão Mac OS X Server do MySQL para o MySQL PKG oficial, você também deve converter a tabela de privilégios do MySQL existente para o formato atual, porque alguns novos privilégios de segurança foram adicionados. Veja Seção 2.5.6 [Upgrading-grant-tables], Página 130.

Se você preferisse iniciar automaticamente o MySQL durante o boot do sistema, você também precisa instalar o MySQL Startup Item. A partir do MySQL 4.0.15, ele é parte do disco de instalação do Mac OS X como um pacote de instalação separado. Simplesmente de um duplo clique no ícone `MySQLStartupItem.pkg` e siga as instruções para instalá-lo.

Note que isto só precisa ser feito uma vez! Não há necessidade de se instalar o Startup Item toda vez que se atualizar o pacote do MySQL.

Devido a um erro no instalador de pacotes do Mac OS X, algumas vezes você pode ver a mensagem de erro `You cannot install this software on this disk. (null)` no diálogo de seleção do disco de destino. Se este erro ocorrer, simplesmente clique no botão `Go Back` uma vez para retornar a tela anterior. Agora clique em `Continue` para avançar para a seleção do disco de destino novamente - agora você deve estar apto a escolher o disco destino corretamente. Nós informamos este erro a Apple e eles estão investigando este problema.

O Startup Item será instalado em '`/Library/StartupItems/MySQL`'. Ele adiciona uma variável `MYSQLCOM=-YES-` ao arquivo de configuração do sistema ('`/etc/hostconfig`'). Se você desejasse diasbilitar a inicialização automática do MySQL, simplesmente altere o valor desta variável para `MYSQLCOM=-NO-`.

No Mac OS X Server, o script de instalação do Startup Item disabilitará automaticamente a inicialização da instalação padrão do MySQL alterando a variável `MYSQL` em '`/etc/hostconfig`' para `MYSQL=-NO-`. Isto é para evitar conflitos na inicialização. No entanto, ele não desliga um servidor MySQL ajá em execução.

Depois da instalação, você pode iniciar o MySQL executando os seguintes comandos em um janela de terminal. Note qye você preceisa ter privilégios de administrador para realizar esta tarefa.

Se você tiver instalado o Startup Item:

```
shell> sudo /Library/StartupItems/MySQL/MySQL start
(Enter your password, if necessary)
(Press Control-D or enter "exit" to exit the shell)
```

Se você não tiver instalado o Startup Item, digite a seguinte sequência de comandos:

```
shell> cd /usr/local/mysql
shell> sudo ./bin/mysqld_safe
(Enter your password, if necessary)
(Press Control-Z)
shell> bg
(Press Control-D or enter "exit" to exit the shell)
```

Agora você deve conseguir se conectar ao servidor MySQL, ex.: executando '`/usr/local/mysql/bin/mysql`'

Se você instalar o MySQL pela primeira vez, **lembre-se de consigurar uma senha para o usuário `root` do MySQL!**

Isto é feito com os seguintes comandos:

```
/usr/local/mysql/bin/mysqladmin -u root password <password>
/usr/local/mysql/bin/mysqladmin -u root -h `hostname` password <password>
```

Por favor, tenha certeza que o comando `hostname` na segunda linha está entre **crases** (`), assim a shell pode substituí-la com a saída deste comando (o nome da máquina deste sistema)!

Você também pode querer adicionar aliases ao seu arquivo de resursos do sheel para acessar `mysql` e `mysqladmin` da linha de comando:

```
alias mysql '/usr/local/mysql/bin/mysql'
alias mysqladmin '/usr/local/mysql/bin/mysqladmin'
```

De forma alternativa, você pode simplesmente adicionar `/usr/local/mysql/bin` a sua variável de ambiente `PATH`, ex.: adicionando o seguinte ao arquivo '`$HOME/.tcshrc`':

```
setenv PATH ${PATH}:/usr/local/mysql/bin
```

Note que instalar um novo MySQL PKG não remove o diretório de uma instalação mais antiga. Infelizmente o Mac OS X Installer ainda não oferece a funcionalidade exigida para atualizar apropriadamente pacotes instalados anteriormente.

Depois de copiar os arquivos de banco de dados do MySQL sobre os da versão anterior e inicializar o nova versão com sucesso, você deve remover os arquivos da instalação antiga para economizar espaço em disco. Adicionalmente você também deve remover versões mais antigas do diretório do Package Receipt localizados em '`/Library/Receipts/mysql-<version>.pkg`'.

2.1.4 Instalando o MySQL no NetWare

A partir da versão 4.0.11, o MySQL está disponível para a Novell NetWare na forma de pacote do binário. Para servir o MySQL, o servidor NetWare deve suprir estas exigências:

- NetWare versão 6.5, ou NetWare 6.0 com Support Pack 3 instalado (Você pode obtê-lo em `http://support.novell.com/filefinder/13659/index.html`). O sistema deve obedecer as exigências mínimas da Naveel para executar a respectiva versão do NetWare.
- Os dados do MySQL, assim com os seus binários, devem ser instalados em um volume NSS; volumes tradicionais não são suportados.

O pacote binário para o NetWare pode ser obtido em `http://www.mysql.com/downloads/`.

Se você estiver executando o MySL no NetWare 6.0, sugerimos que você utilize a opção `--skip-external-locking` na linha de comando. Também será necessário utilizar `CHECK TABLE` e `REPAIR TABLE` em vez de `myisamchk`, porque `myisamchk` faz uso de lock externo. Lock externo possui problemas com NetWare 6.0; o problema foi eliminado no NetWare 6.5.

2.1.4.1 Instalando o MySQL para Binários do NetWare

1. Se você estiver atualizando de um instaação anterior, para o servidor MySQL. Isto é feito a partir do console do servidor, usando:

   ```
   SERVER:  mysqladmin -u root shutdown
   ```

2. Conecte-se no servidor alvo a partir de uma máquina cliente com acesso ao local onde você instalará o MySQL.
3. Extraia o pacote zip binário em seu servidor. Tenha certeza de permitir que os caminhos no arquivo zip sejam usados. É seguro simplesmente extrair os arquivos para '`SYS:\`'.

   Se você estiver atualizando de uma instalando anterior, você pode precisar copiar os diretórios de dados (ex.: '`SYS:MYSQL\DATA`') agora, assim como '`my.cnf`' se você o tiver costumizado. Você pode então deletar a cópia antiga do MySQL.
4. Você pode desejar renomear o diretório para algo mais consistente e fácil de usar. Recomendamos usar o '`SYS:MYSQL`'; exemplos no manual o usarão para se referir ao diretório de instalação em geral.
5. No console do servidor, adicione um caminho de busca no diretório contendo os NLMs do MySQL. Por exemplo:

   ```
   SERVER:  SEARCH ADD SYS:MYSQL\BIN
   ```

6. Instale o banco de dados inicial, se necessário, digitando `mysql_install_db` no console do servidor.
7. Inicie o servidor MySQL usando `mysqld_safe` no console do servidor.

8. Para finalizar a instalação, você também deve adicionar os seguintes comandos ao `autoexec.ncf`. Por exemplo, se sua instalação do MySQL está em '`SYS:MYSQL`' e você quiser que o MySQL inicie automaticamente, você pode adicionar estas linhas:

   ```
   #Starts the MySQL 4.0.x database server
   SEARCH ADD SYS:MYSQL\BIN
   MYSQLD_SAFE
   ```

   Se você estiver usando NetWare 6.0, você deve adicionar o parâmetro `--skip-external-locking`:

   ```
   #Starts the MySQL 4.0.x database server
   SEARCH ADD SYS:MYSQL\BIN
   MYSQLD_SAFE --skip-external-locking
   ```

Se houver uma instalação existente do MySQL no servidor, verifique a existencia de comandos de inicialização do MySQL em `autoexec.ncf`, e edite ou delete-os se necessário.

2.2 Detalhes Gerais de Instalação

2.2.1 Como obter o MySQL

Confira a homepage da MySQL homepage (`http://www.mysql.com/`) para informações sobre a versão atual e para instruções de download.

Nosso principal espelho de download está localizado em: `http://mirrors.sunsite.dk/mysql/`.

Para uma lista atualizada completa dos mirrors de download da MySQL, veja `http://www.mysql.com/downloads/mirrors.html`. Você também encontrará informação sobre como se tornar um mirror do MySQL e como relatar um mirror ruim ou desatualizado.

2.2.2 Verificando a Integridade do Pacote Usando `MD5 Checksums` ou `GnuPG`

Depois de fazer o download do pacote MySQL que serve às suas necessidades e antes de tentar instalá-lo, você deve ter certeza de que ele esta intacto e não foi manipulado.

A MySQL AB oferece dois tipos de verificação de integridade: `MD5 checksums` e assinaturas criptografadas usando `GnuPG`, o `GNU Privacy Guard`.

Verificando o `MD5 Checksum`

Depois de fazer o download do pacote, você deve verificar se o MD5 checksum corresponde a aquele fornecido na página de download do MySQL. Cada pacote tem um checksum individual, que você pode verificar com o seguinte comando:

```
shell> md5sum <pacote>
```

Note que nem todos os sistemas operacionais suportam o comando `md5sum` - em alguns ele é simplesmente chamado `md5`, outros não o possuem. No Linux, ele é parte do pacote `GNU Text Utilities`, que está disponível para uma grande faixa de plataformas. Você

pode fazer o download do código fonte em `http://www.gnu.org/software/textutils/`. Se você tiver o `OpenSSL` instalado, você também pode usar o comando `openssl md5 <pacote>`. Uma implementação do comando `md5` para DOS/Windows está disponível em `http://www.fourmilab.ch/md5/`.

Exemplo:

```
shell> md5sum mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz
155836a7ed8c93aee6728a827a6aa153
                mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz
```

Você deve verificar se o resultado do checksum corresponde a aquele impresso no página de download logo abaixo do respectivo pacote.

A maioria do sites mirrors também oferecem um arquivo chamado '`MD5SUMS`', que também inclui o MD5 checksums para todos os arquivos incluídos no diretório '`Downloads`'. Note no entanto que é muito fácil de modificar este arquivo e ele não é um método muito confiável. Caso esteja em dúvida, você deve consultar diferentes sites mirroers e comparar os resultados.

Verificação de Assinatura Usando `GnuPG`

Um método de verificação de integridade de um pacote mais confiável é o uso de assinaturas criptografadas. A MySQL AB usa o `GNU Privacy Guard` (`GnuPG`), uma alternativa `Open Source` para o bem conhecido `Pretty Good Privacy` (`PGP`) de Phil Zimmermann. Veja `http://www.gnupg.org/` and `http://www.openpgp.org/` para mais informações sobre `OpenPGP`/`GnuPG` e como obter e instalar o `GnuPG` em seus sistema. A maioria das distribuições de Linux já vêm com o `GnuPG` instalado por padrão.

A partir do MySQL 4.0.10 (Fevereiro de 2003), a MySQL AB começou a assinar o seus pacotes de download com `GnuPG`. Assinaturas criptografadas são um método bem mais confiável de verificação da integridade e autenticidade de um arquivo.

Para verificar a assinatura de um pacote específico, você primeiro precisa obtter uma cópia da chave pública GPG da MySQL AB (`build@mysql.com`). Você também pode cortá-la e colá-la diretamente daqui ou obtê-la em `http://www.keyserver.net/`.

```
Key ID:
pub  1024D/5072E1F5 2003-02-03
     MySQL Package signing key (www.mysql.com) <build@mysql.com>
Fingerprint: A4A9 4068 76FC BD3C 4567  70C8 8C71 8D3B 5072 E1F5

Public Key (ASCII-armored):

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.0.6 (GNU/Linux)
Comment: For info see http://www.gnupg.org

mQGiBD4+owwRBAC14GIfUfCyEDSIePvEW3SAFUdJBtoQHH/nJKZyQT7h9bPlUWC3
RODjQReyCITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPkBdCk96+OmSLN9brZ
fw2vOUgCmYv2hWOhyDHuvYlQA/BThQoADgj8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3
BqOxRznNCRCRxAuAuVztHRcEAJooQK1+iSiunZMYD1WufeXfshc57S/+yeJkegNW
```

```
hxwR9pRWVArNYJdDRT+rf2RUe3vpquKNQU/hnEIUHJRQqYHo8gTxvxXNQc7fJYLV
K2HtkrPbP72vwsEKMYhhr0eKCbtLGfls9krjJ6sBgACyP/Vb7hiPwxh6rDZ7ITnE
kYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgtobZX9qnrAXw+uNDI
QJEXM6FSbi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPAnWqcyefep
rv0sxxqUEMcM3o7wwgfN83POkDasDbs3pjwPhxvhz6//62zQJ7Q7TXlTUUwgUGFj
a2FnZSBzaWduaW5nIGtleSAod3d3Lm15c3FsLmNvbSkgPGJ1aWxkQG15c3FsLmNv
bT6IXQQTEQIAHQUCPj6jDAUJCWYBgAULBwoDBAMVAwIDFgIBAheAAAoJEIxxjTtQ
cuH1cY4AnilUwTXn8MatQOiG0a/bPxrvK/gCAJ4oinSNZRYTnblChwFaazt7PF3q
zIhMBBMRAgAMBQI+PqPRBYMJZgC7AAoJEElQ4SqycpHyJOEAn1mxHijft00bKXvu
cSo/pECUmppiAJ41M9MRVj5VcdH/KN/KjRtW6tHFPYhMBBMRAgAMBQI+QoIDBYMJ
YiKJAAoJELb1zU3GuiQ/lpEAoIhpp6BozKI8p6eaabzF5MlJH58pAKCu/ROofK8J
Eg2aLos+5zEYrB/LsrkCDQQ+PqMdEAgA7+GJfxbMdY4wslPnjH9rF4N2qfWsEN/l
xaZoJYc3a6M02WCnHl6ahT2/tBK2w1QI4YFteR47gCvtgb6O1JHffOo2HfLmRDRi
Rjd1DTCHqeyX7CHhcghj/dNRlW2ZOl5QFEcmV9UOVhp3aFfWC4Ujfs3LU+hkAWzE
7zaD5cH9J7yv/6xuZVw411x0h4UqsTcWMu0iM1BzELqX1DY7LwoPEb/O9Rkbf4fm
Le11EzIaCa4PqARXQZc4dhSinMt6K3X4BrRsKTfozBu74F47D8Ilbf5vSYHbuE5p
/1oIDznkg/p8kW+3FxuWrycciqFTcNz215yyX39LXFnlLzKUb/F5GwADBQf+Lwqq
a8CGrRfsOAJxim63CHfty5mUc5rUSnTslGYEIOCR1BeQauyPZbPDsDD9MZ1ZaSaf
anFvwFG6Llx9xkU7tzq+vKLoWkm4u5xf3vn55VjnSd1aQ9eQnUcXiL4cnBGoTbOW
I39EcyzgslzBdC++MPjcQTcA7p6JUVsP6oAB3FQWg54tuUo0Ec8bsM8b3Ev42Lmu
QT5NdKHGwHsXTPtl0klk4bQk4OajHsiy1BMahpT27jWjJlMiJc+IWJ0mghkKHt92
6s/ymfdf5HkdQ1cyvsz5tryVI3Fx78XeSYfQvuuwqp2H139pXGEkg0n6KdUOetdZ
Whe70YGNPw1yjWJT1IhMBBgRAgAMBQI+PqMdBQkJZgGAAAoJEIxxjTtQcuH17p4A
n3r1QpVC9yhnW2cSAjq+kr72GX0eAJ4295kl6NxYEuFApmr1+0uUq/SlsQ==
=YJkx
-----END PGP PUBLIC KEY BLOCK-----
```

Você pode importar esta chave em seu pasta de chaves publicas `GPG` usando `gpg --import`. Veja a documentação de `GPG` para mais informações de como trabalhar com chaves públicas.

Depois de fazer o download e importar a chave publica criada, faça o download do pacote MySQL desejado e da assinatura correspondente, que também está disponível na página de download. A assinatura tem a extensão '`.asc`'. Por exemplo, a assinatura de '`mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz`' seria '`mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz.asc`'. Tenha certeza que ambos os arquivos estão armazenados no mesmo diretório e então execute o seguinte comando para verificar a assinatura para este arquivo:

```
shell> gpg --verify <package>.asc

Exemplo:

shell> gpg --verify mysql-standard-4.0.10-gamma-pc-linux-i686.tar.gz.asc
gpg: Warning: using insecure memory!
gpg: Signature made Mon 03 Feb 2003 08:50:39 PM MET using DSA key ID 5072E1F5
gpg: Good signature from
     "MySQL Package signing key (www.mysql.com) <build@mysql.com>"
```

A mensagem "Good signature" indica que está tudo certo.

Verificando Assinatura Usando RPM

Para pacotes RPM, não há assinaturas separadas - pacotes RPM atualmente têm uma assinatura GPG incluída e MD5 checksum. Você pode verificá-los executando o seguinte comando:

```
shell> rpm --checksig <package>.rpm

Exemplo:

shell> rpm --checksig MySQL-server-4.0.10-0.i386.rpm
MySQL-server-4.0.10-0.i386.rpm: md5 gpg OK
```

Nota: Se você estiver usando RPM 4.1 e ele reclamar sobre (GPG) NOT OK (MISSING KEYS: GPG#5072e1f5) (mesmo se você a importou para detro de sua pasta de chaves publicas GPG), você precisa importá-las para dentro de sua pasta de chaves RPM primeiro. RPM 4.1 não utiliza mais ias suas pastas de chaves GPG (e o próprio GPG), mas mantém sua própria pasta de chaves (porque ele é um aplicativo do sistema e a pasta de chaves públicas do GPG é um arquivo específico do usuário). Para importar a chave pública do MySQL em uma pasta de chaves RPM, use os seguintes comandos:

```
shell> rpm --import <pubkey>

Exemplo:

shell> rpm --import mysql_pubkey.asc
```

Caso você note que as assinaturas MD5 checksum ou GPG não coincidem, tente primeiro fazer o download do pacote respectivo mais uma vez, talvez de outro site mirror. Se você não obter sucesso na verificação da integridade do pacote repetidas vezes, notifique-nos sobre tais incidentes incluindo o nome completo do pacote e o site que você tem utilizado para fazer o download pelos emails webmaster@mysql.com ou build@mysql.com.

2.2.3 Sistemas Operacionais suportados pelo MySQL

Nós ulitizamos o GNU Autoconf, para que seja possível portar o MySQL para todos sistemas operacionais modernos com threads Posix funcionando e um compilador C++. (Para compilar somente o código cliente, um compilador C++ é necessário mas threads não.) Nós mesmos usamos e desenvolvemos o software primeiramente no Linux (SuSE e red Hat), FreeBSD e Sun Solaris (Versões 8 e 9).

Perceba que para alguns sistemas operacionais, o suporte nativo a thread funciona somente nas últimas versões. O MySQL compila com sucesso nas seguintes combinações de sistema operacional/pacote de thread:

- AIX 4.x com threads nativas. Veja Seção 2.6.6.4 [IBM-AIX], Página 155.
- Amiga.
- BSDI 2.x com o pacote incluído MIT-pthreads. Veja Seção 2.6.4.5 [BSDI], Página 151.
- BSDI 3.0, 3.1 e 4.x com threads nativas. Veja Seção 2.6.4.5 [BSDI], Página 151.
- SCO OpenServer with a recent port of the FSU Pthreads package. Veja Seção 2.6.6.9 [SCO], Página 161.

- SCO UnixWare 7.0.1. Veja Seção 2.6.6.10 [SCO Unixware], Página 163.
- DEC Unix 4.x com threads nativas. Veja Seção 2.6.6.6 [Alpha-DEC-UNIX], Página 157.
- FreeBSD 2.x com o pacote incluído MIT-pthreads. Veja Seção 2.6.4.1 [FreeBSD], Página 148.
- FreeBSD 3.x e 4.x com threads nativas. Veja Seção 2.6.4.1 [FreeBSD], Página 148.
- FreeBSD 4.x com Linuxthreads. Veja Seção 2.6.4.1 [FreeBSD], Página 148.
- HP-UX 10.20 com o pacote incluído MIT-pthreads ou DCE threads. Veja Seção 2.6.6.2 [HP-UX 10.20], Página 153.
- HP-UX 11.x com as threads nativas. Veja Seção 2.6.6.3 [HP-UX 11.x], Página 154.
- Linux 2.0+ com LinuxThreads 0.7.1+ ou `glibc` 2.0.7+. Veja Seção 2.6.2 [Linux], Página 137.
- Mac OS X Server. Veja Seção 2.6.5 [Mac OS X], Página 152.
- NetBSD 1.3/1.4 Intel e NetBSD 1.3 Alpha (Necessita GNU make). Veja Seção 2.6.4.2 [NetBSD], Página 150.
- Novell NetWare 6.0. Veja Seção 2.6.8 [Novell NetWare], Página 164.
- OpenBSD > 2.5 com threads nativas. OpenBSD < 2.5 com o pacote incluído MIT-pthreads . Veja Seção 2.6.4.3 [OpenBSD], Página 150.
- OS/2 Warp 3, FixPack 29 e OS/2 Warp 4, FixPack 4. Veja Seção 2.6.7 [OS/2], Página 163.
- SGI Irix 6.x com threads nativas. Veja Seção 2.6.6.8 [SGI-Irix], Página 160.
- Solaris 2.5 e superior com threads nativas nas plataformas SPARC e x86. Veja Seção 2.6.3 [Solaris], Página 144.
- SunOS 4.x com o pacote incluído MIT-pthreads. Veja Seção 2.6.3 [Solaris], Página 144.
- Tru64 Unix
- Windows 9x, Me, NT, 2000 e XP. Veja Seção 2.6.1 [Windows], Página 133.

Perceba que nem todas as plataformas são apropriadas para executar o MySQL. Os seguintes fatores determinam se uma certa plataforma é apropriada para uma missão crítica pesada:

- Estabilidade geral da biblioteca thread. Uma plataforma pode ter excelente reputação, entretanto, se a biblioteca thread é instável no código que é usado pelo MySQL, mesmo se todo o resto for perfeito, o MySQL irá ser tão estável quanto a biblioteca thread.
- A habilidade do kernel e/ou a biblioteca thread tirar vantagem do **SMP** em sistemas multi-processados. Em outras palavras, quando um proceesso cria uma thread, deve ser possível para aquela thread executar em uma CPU diferente que o processo original.
- A habilidade do kernel e/ou a biblioteca thread executar várias threads que adiquire/libera um bloqueio mutex sobre uma pequena região crítica frequentemente sem trocas de contexto excessivos. Em outras palavras, se a implementação de `pthread_mutex_lock()` requisitar a CPU muito rapidamente, isto irá afetar o MySQL tremendamente. Se esse detalhe não estiver sendo cuidado, adicionar CPUs extras podem deixar o MySQL mais lento.
- Estabilidade e performance geral do sistema de arquivos.
- Habilidade do sistema de arquivos em lidar com arquivos grandes de forma eficiente, se suas tabelas forem grandes.

- Nosso nível de experiência aqui na MySQL AB com a plataforma. Se nós conhecemos bem uma plataforma, introduzimos otimizações/correçoes específicas para ela habilitadas na hora da compilação. Nós também podemos fornecer conselhos sobre como configurar seu sistema otimizadamente para o MySQL.
- O volume de testes feitos internamente de configurações similares.
- O número de usuários que tem executado o MySQL com sucesso naquela plataforma em configurações similares. Se esse número for alto, as chances de se ter alguma surpresa específica da plataforma fica muito menor.

Baseado nos critérios acima, as melhores plataformas para a execução do MySQL até este ponto são o x86 com SuSe Linux 8.2, kernel 2.4 e ReiserFS (ou qualquer distribuição Linux similar) e Sparc com Solaris (2.7-9). FreeBSD vem em terceiro, mas realmente temos esperanças que ele irá se unir ao clube dos tops uma vez que a biblioteca thread está melhorando. Nós também acreditamos que em certo ponto iremos estar aptos para incluir todas as outras plataformas em que o MySQL compila e executa, mas não tão bem e com o mesmo nível de estabilidade e performance, na categoria superior. Isto necessitará de algum esforço da nossa parte em cooperação com os desenvolvedores dos componentes do Sistema Operacional/Biblioteca que o MySQL depende. Se você tiver interesse em melhorar algum de nossos componentes, está em uma posição para influenciar seu desenvolvimento, e precisa de instruções mais detalhadas sobre o que o MySQL necessita para uma melhor execução, envie um e-mail para lista de email "insternals" do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33.

Por favor, perceba que a comparação acima não é para dizer que um SO é melhor ou pior que o outro em geral. Nós estamos falando sobre a escolha de um SO para um propósito dedicado: executar o MySQL, e comparamos as plataformas levando isto em consideração. Desta forma, o resultado desta comparação seria diferente se nós incluíssemos mais detalhes. E em alguns casos, a razão de um SO ser melhor que o outro pode ser simplesmente porque colocamos mais esforço nos testes e otimização para aquela plataforma em particular. Estamos apenas colocando nossas observações para ajudá-lo na decisão de qual plataforma usar o MySQL na sua configuração.

2.2.4 Qual versão do MySQL deve ser usada

A primeira decisão a ser feita é se você deve usar a última versão de desenvolvimento ou a última versão estável:

- Normalmente, se você estiver usando o MySQL pela primeira vez ou tentando portá-lo para algum sistema em que não exista distribuição binária, recomendamos o uso da versão estável (atualmente Versão 5.0.0-alpha). Repare que todos os lançamentos do MySQL são conferidos com os testes comparativos de performance e um conjunto extenso de testes antes de cada lançamento.
- Senão, caso você esteja trabalhando com um antigo sistema e quiser atualizá-lo, mas não que correr o risco com uma atualização sem correções, você deve faze-lo do mesmo ramo que você está usando (onde aenas o último número da versão é mais novo que o seu). Nós temos tentado corrigir somente erros fatais e torná-los menores, com alterações relativamente seguras para aquela versão.

A segunda decisão a ser feita é se você deseja usar uma distribuição fonte ou binária. Na maioria dos casos provavelmente você deverá usar a distribuição binária, se alguma existir para sua plataforma, será normalmente muito mais fácil para instalar do que a distribuição em código fonte.

Nos seguites casos você provavelmente será mais bem servido com uma instalação baseada em código fonte:

- Se você desejar instalar o MySQL em algum lugar expecífico. (O padrão das distribuições binárias é estar"pronto para rodar" em qualquer lugar, mas talvez você deseje ainda mais flexibilidade).
- Para estar apto e satisfazer diferentes requisições dos usuários, estaremos fornecendo duas versões binárias diferentes; Uma compilada com os manipuladores de tabelas não transacionais (um binário rápido e pequeno) e um configurado com as mais importantes opções extendidas como tabelas transacionais. Ambas versões são compiladas da mesma distribuição fonte. Todos clientes `MySQL` nativos pode conectar com ambas versões do MySQL.

  A distribuição binária extendida é marcada com o sufixo `-max` e é configurada com as mesmas opções de `mysqld-max`. Veja Seção 4.8.5 [`mysqld-max`], Página 343.

  Se você deseja usar o RPM `MySQL-Max`, primeiramente você deve instalar o RPM `MySQL-server` padrão.
- Se você deseja configurar `mysqld` com alguns recursos extras que NÃO estão nas distribuições binárias. Segue abaixo a lista das opções extras mais comuns que você pode querer usar:
  - `--with-innodb`
  - `--with-berkeley-db` (padrão para o MySQL 4.0 e seguintes)
  - `--with-raid` (não disponível para todas as plataformas)
  - `--with-libwrap`
  - `--with-named-z-lib` (Isto é feito para alguns dos binários)
  - `--with-debug[=full]`
- A distribuição binária padrão é normalmente compilada com suporte para todos conjuntos de caracteres e deve funcionar em uma variedade de processadores para a mesma família do processador.

  Se você precisar de um servidor MySQL mais rápido você pode querer recompilá-lo com suporte para somente o conjunto de caracteres que você precisa, usar um compilador melhor (como `pgcc`) ou usar opções de compiladores para usar otimizações para seu processador.
- Se você encontrar um erro e relatá-lo para o time de desenvolvimento do MySQL você provavelmente receberá um patch que será necessário aplicá-lo para a distribuição fonte para ter o bug corrigido.
- Se você deseja ler (e/ou modificar) o código C e C++ que é o MySQL, você pode obter uma distribuição fonte. O código fonte é sempre o manual final. Distribuições fontes também contem mais testes e exemplos que as distribuições binárias.

O esquema de nomes do MySQL usa números de versões que consistem de tres números e um sufixo. Por exemplo, um nome de lançamento como `mysql-4.1.0-alpha` é interpretado da seguinte maneira:

- O primeiro número (`4`) é a versão principal e também descreve o formato dos arquivos. Todas releases da Versão 4 tem o mesmo formato de arquivo.
- O segundo número (`1`) é o nível da distribuição.
- O terceiro número (`0` é o número da versão do nível de distribuição. Este é incrementado para cada nova distribuição. Normalmente você desejará a última versão para o nível de publicação que tiver escolhido.
- O sufixo (`alpha`) indica o nível de estabilidade da versão. Os possíveis sufixo são:
  - `alpha` indica que a versão contém grandes seções de novos códigos que não foram 100% testados. Bugs conhecidos (normalmente não tem nenhum) devem estar documentados na seção News. Veja Apêndice D [News], Página 957. Existem também novos comandos e extensões na maioria das publicações alpha. Desenvolvimento ativo que podem envolver maiores alterações no código pode ocorrer numa versão alpha, mas tudo será testado antes de fazer a publicação. Não podem existir erros conhecidos em nenhuma publicação do MySQL.
  - `beta` significa que todo o novo código foi testado. Não serão adicionados novos recursos que podem causar algum tipo de corrompimento. Não deve existir bugs conhecidos. Uma alteração de versão de alpha para beta ocorre quando não existir nenhum relato de erro fatal com uma versão alpha por pelo menos um mês e não planejarmos adicionar nenhum recurso que pode deixar algum antigo comando menos confiável.
  - `gamma` é o beta que já tem sido usado a algum tempo e parece funcionar bem. Apenas pequenas correções são adicionadas. Isto é o que muitas empresas chamam de release.
  - Se não existir um sufixo, significa que esta versão já está sendo executada há algum tempo em diferentes locais sem relatos de erros além dos específicos de certas plataformas. Somente correções de erros críticos são adicionados ao release. Isto é o que chamamos de uma distribuição estável.

No processo de desenvolvimento do MySQL, várias versões coexistem e estão em um estágio diferente. Naturalmente, correções de erros relevantes de uma série anterior são propagados.

- Para a antiga série `3.23` estável/de produção, novas versões só são liberadas para erros críticos.
- A série atual (`4.0`) é de qualidade estável/produção. Nenhum novo recurso que possa influenciar a estabilidade do código é adicionado.
- No ramo alpha `4.1` principal, novos recursos são adicionados. Fontes e binários estão disponíveis para uso e teste em sistemas de desenvolvimento.
- O ramo de desenvolvimento `5.0` só está disponível para a árvore do BitKeeper.

Todas as versões do MySQL funcionam sobre nossos testes padrões e comparativos para garantir que eles são relativamente seguros para o uso. Como os testes padrões são extendidos ao longo do tempo para conferir por todos os bugs antigos encontrados, o pacote de testes continua melhorando.

Perceba que todas publicações de versões foram testadas pelo menos com:

Um pacote de testes interna
: Faz parte de um sistema de produção para um cliente. Ela tem diversas tabelas com centenas de megabytes de dados.

O diretório '`mysql-test`' contém um conjunto extensivo de casos de teste. Nós executamos estes testes para cada servidor binário.

O pacote de comparativos da MySQL
: Este executa uma série de consultas comuns. É também um teste para ver se o último conjunto de otimizações fez o código mais rápido. Veja Seção 5.1.4 [MySQL Benchmarks], Página 421.

O teste `crash-me`
: Este tenta determinar quais recursos o banco de dados suporta e quais são suas capacidades e limitações. Veja Seção 5.1.4 [MySQL Benchmarks], Página 421.

Outro teste é que nós usamos a versão do MySQL mais nova em nosso ambiente de produção interna, em pelo menos uma máquina. Nós temos mais de 100 gigabytes de dados com que trabalhar.

2.2.5 Layouts de Instalação

Esta seção descreve o layout padrão dos diretórios criados pela instalção das distribuições binária e fonte.

Uma distribuição binária é instalada descompactando-a no local de instalação de sua escolha (tipicamente '`/usr/local/mysql`') e cria os seguintes diretórios nesses locais:

Diretório	**Conteúdo do diretório**
'`bin`'	Programas clientes e o servidor `mysqld`
'`data`'	Arquivos Log, bancos de dados
'`docs`'	Documentação, Log de alterações
'`include`'	Arquivos de cabeçalho (headers)
'`lib`'	Bibliotecas
'`scripts`'	`mysql_install_db`
'`share/mysql`'	Arquivos de mensagem de erro
'`sql-bench`'	Benchmarks - testes comparativos

Uma distribuição baseada em código fonte é instalada depois de você configurá-la e compilá-la. Por padrão, a instalação copia os arquivos em '`/usr/local`', nos seguintes subdiretórios:

Diretório	**Conteúdo do diretório**
'`bin`'	Programas clientes e scripts
'`include/mysql`'	Arquivos de cabeçalho (headers)
'`info`'	Documentação no formato Info
'`lib/mysql`'	Bibliotecas
'`libexec`'	O servidor `mysqld`
'`share/mysql`'	Arquivos com mensagens de erros
'`sql-bench`'	Benchmarks e o teste `crash-me`
'`var`'	Bancos de dados e arquivos log

Dentro de um diretório de instalação, o layout de uma instalação baseada em fontes diferencia de uma instalação binária nas seguintes formas:

- The `mysqld` server is installed in the '`libexec`' directory rather than in the '`bin`' directory.
- The data directory is '`var`' rather than '`data`'.

- `mysql_install_db` is installed in the '`/usr/local/bin`' directory rather than in '`/usr/local/mysql/scripts`'.
- The header file and library directories are '`include/mysql`' and '`lib/mysql`' rather than '`include`' and '`lib`'.

You can create your own binary installation from a compiled source distribution by executing the script '`scripts/make_binary_distribution`'.

2.2.6 Como e quando as atualizações são lançadas?

O MySQL está evoluindo muito rapidamente na MySQL AB e nós queremos compartilhar isto com outros usuários MySQL. Sempre que temos alguns recursos úteis que outros acham necessáio, tentamos fazer um release.

Também tentamos ajudar usuários que solicitam recursos que são de fácil implementação. Tomamos notas do que nossos usuários licenciados gostariam de ter,especialmente do que nossos clientes com suporte extendido desejam e tentamos ajudá-los.

Não existe uma real necessidade para baixar uma nova release. A seção News irá dizer se a nova versão tem alguma coisa que você precisa. Veja Apêndice D [News], Página 957.

Usamos a seguinte polítca quando estamos atualizando o MySQL:

- Para cada pequena atualização, o último número na versão é incrementado. Quando tiver um maior número de novos recursos ou menor incompatibilidade com versões antigas, o segundo número na versão é incrementado. Quando o formato de arquivo altera, o primeiro número é aumentado.
- Versões estáveis testadas aparecem na média de uma a duas vezes por ano, mas se pequenos bugs são encontrados, uma versão será lançada apenas com as correções dos erros.
- Releases funcionais aparecem na média a cada 4-8 semanas.
- Distribuições binárias para algumas plataformas será feita por nós somente para releases mais importantes. Outras pessoas podem fazer distribuições binárias para outros sistemas mas provavelmente com menos frequencia.
- Nós normalmente disponibilizamos os patches logo que localizamos e corrigimos pequenos bugs. Eles normalmente são imediatamente disponibilizados em nosso repositório publico do BitKeeper. Eles serão incluídos na próxima distribuição.
- Para bugs não críticos, mas irritantes, disponibilizamos patches se eles são enviados para nós. De qualquer forma, iremos combinar vários deles em um patch maior.
- Se existitr, por algum motivo, um bug fatal numa versão criaremos uma nova release o mais cedo possível. Gostaríamos que outras empresas fizessem isto também.

A versão estável atual é a 3.23; nós já mudamos o desenvolvimento em atividade para a versão 4.0. Bugs contiuarão a ser corrigidos na versão estável. Não acreditamos em um congelamento completo, pois isto abandona a correções de bugs e coisas que "devem ser feitas." "Alguma coisa congelada" significa que talvez possamos adicionar pequenas coisas que "com certeza não afetará nada que já esteja funcionando."

O MySQL usa um esquema de nomes um pouco diferente da maioria dos outros produtos. Em geral é relativamente seguro utilizar qualquer versão que tenha sido lançado a algumas

semanas e que não tenham sido sustituída por uma nova versão. Veja Seção 2.2.4 [Qual versão], Página 80.

2.2.7 Filosofia das Distribuições - Nenhum Bug Conhecidos nas Distribuições

Colocamos muito tempo e esforço em tornar nossas distribuições livres de erros. Pelo nosso conhecimento, não liberamos uma única versão do MySSQL com qualquer erro `conhecido` 'fatal'.

Um erro fatal é algo que faz o MySQL falhar com o uso normal, traz respostas erradas para consultas normais ou tem um problema de segurança.

Nós temos documentados todos os problemas conhecidos, bugs e assuntos que são dependentes das decisões de projeto. Veja Seção 1.8.6 [Bugs], Página 54.

Nosso objeto é corrigir tudo que é possível, mas sem correr o risco de tornarmos uma versão menos estável. Em certos casos, isto significa que podemos corrigir um problema na(s) versão(ões) de desenvolvimento, mas não o corrigirmos na versão estável (produção). Naturalmente, documentamos tais problemas para que o usuários esteja ciente.

Aqui está um descrição de como nosso processo de contrução funciona:

- Monitoramos erros de nossa lista de suporte ao cliente, a lista de email externa do MySQL e o banco de dados de bugs em `http://bugs.mysql.com/`.
- Todos os erros relatados em versões usadas são inseridos nio banco de dados de bugs.
- Quando corrigimos um erro, sempre tentamos fazer um caso de teste para ele e incluí-lo em nosso sistema de teste para assegurar que o erro nunca retornará. (Cerca de 90% de todos os erros corrigidos têm um caso de teste.)
- Também criamos casos de teste para todos os novos recursos adicionados ao MySQL.
- Antes de começarmos a fazer uma nova distribuição do MySQL, asseguramos que todos os erros repetitíveis relatados para a versão do MySQL (3.23.x, 4.0.x, etc) estão corrigidos. Se algo for impossível de corrigir (devido a alguma decisão de projeto interno no MySQL), documentaremos isto no manual. Veja Seção 1.8.6 [Bugs], Página 54.
- Nós fazemos uma construção para todas as plataformas para as quais suportamos binários (mais de 15 plataformas) e executamos nosso pacote de teste e benchmarks para todas elas.
- Não publicaremos um binário para uma plataforma na qual os testes do pacote de benchmarks falharam. Se for um erro geral na fonte, o corrigimos e fazemos as contruções mais os teste em todos os sistemas novamente.
- Se nós, durante a o porcesso de construção e teste (que leva de 2 a 3 dias) recebermos um relatório sobre um erro fatal (por exemplo, um que cause um core dump), o corrigiremos e reiniciaremos o processo de construção).
- Depois de publicarmos os binários em `http://www.mysql.com/`, enviamos um email de anúncio nas listas de email `mysql` e `announce`. Veja Seção 1.7.1.1 [Mailing-list], Página 33. A mensagem de anúncio contém uma lista de todas as alterações da distribuição e qualquer problema conhecido com ela. (A seção 'problemas conhecidos' na notas das distribuições só tem sido necessária em algumas da distribuições.)

- Para darmos acesso rapidamente aos nossos usuários dos últimos recursos do MySQL, fazemos uma nova distribuição do MySQL a cada 4-8 semanas. Instantâneos do código finte são contruídos diariamente e estão disponíveios em `http://downloads.mysql.com/snapshots.php`.
- Se, depois da distribuição estar pronta, recebermos qualquer relatório que houve (depois de tudo) qualquer coisa criticamente errada com a construção em uma plataforma específica, corrigiremo-na imediatamente e liberaremos um nova distribuição `'a'` para aquela plataforma. Graças aos nosso grande base de usuários, problemas são encontrados rapidamente.
- Nosso registro para boas distribuições feitas é muito boa. Nas últimas 150 distribuições, tivemos que fazer uma nova contrução para menos de 10 distribuições (em 3 destes casos, o erro era uma biblioteca glibc com problema em uma de nossas máquinas que levamos um longo tempo para encontrar.

2.2.8 Binários MySQL compilados pela MySQL AB

Como um serviço, nós na MySQL AB fornecemos um conjunto de distribuições binárias do MySQL que são compiladas no nosso site ou em sites onde os clientes cordialmente nos dão acesso as suas máquinas.

Em adição aos binários forncedios em formatos de pacotes específicos da plataforma (veja Seção 2.1 [Quick Standard Installation], Página 60), oferecemos distribuições binários para outras plataformas através de arquivos tar compactados (`.tar.gz`).

Estas distribuições são geradas usando o script `Build-tools/Do-compile` que compila o código fonte e cria o arquivo binário em `tar.gz` usando `scripts/make_binary_distribution`. Estes binários são configurados e construídos com os seguintes compiladores e opções.

Binários construídos no sistema de desenvolvimento da MySQL AB:

Linux 2.4.xx x86 com `gcc` 2.95.3

```
CFLAGS="-O2 -mcpu=pentiumpro" CXX=gcc CXXFLAGS="-O2 -
mcpu=pentiumpro -felide-constructors" ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex --enable-thread-safe-client
--enable-local-infile --enable-assembler --disable-shared --with-
client-ldflags=-all-static --with-mysqld-ldflags=-all-static
```

Linux 2.4.xx Intel Itanium 2 com `ecc` (Intel C++ Itanium Compiler 7.0)

```
CC=ecc CFLAGS="-O2 -tpp2 -ip -nolib_inline" CXX=ecc CXXFLAGS="-O2
-tpp2 -ip -nolib_inline" ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex --enable-thread-safe-client
--enable-local-infile
```

Linux 2.4.xx Intel Itanium com `ecc` (Intel C++ Itanium Compiler 7.0)

```
CC=ecc CFLAGS=-tpp1 CXX=ecc CXXFLAGS=-tpp1 ./configure --
prefix=/usr/local/mysql --with-extra-charsets=complex --enable-
thread-safe-client --enable-local-infile
```

Linux 2.4.xx alpha com `ccc` (Compaq C V6.2-505 / Compaq C++ V6.3-006)

```
CC=ccc CFLAGS="-fast -arch generic" CXX=cxx CXXFLAGS="-
fast -arch generic -noexceptions -nortti" ./configure -
```

```
-prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --with-
mysqld-ldflags=-non_shared --with-client-ldflags=-non_shared
--disable-shared
```

Linux 2.4.xx s390 com gcc 2.95.3

```
CFLAGS="-O2" CXX=gcc CXXFLAGS="-O2 -felide-constructors"
./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared
--with-client-ldflags=-all-static --with-mysqld-ldflags=-all-
static
```

Linux 2.4.xx x86_64 (AMD64) com gcc 3.2.1

```
CXX=gcc ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--disable-shared
```

Sun Solaris 8 x86 com gcc 3.2.3

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-
O3 -fno-omit-frame-pointer -felide-constructors -fno-
exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --
localstatedir=/usr/local/mysql/data --libexecdir=/usr/local/mysql/bin
--with-extra-charsets=complex --enable-thread-safe-client
--enable-local-infile --disable-shared --with-innodb
```

Sun Solaris 8 sparc com gcc 3.2

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3
-fno-omit-frame-pointer -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--enable-assembler --with-named-z-libs=no --with-named-curses-
libs=-lcurses --disable-shared
```

Sun Solaris 8 sparc 64bit com gcc 3.2

```
CC=gcc CFLAGS="-O3 -m64 -fno-omit-frame-pointer" CXX=gcc
CXXFLAGS="-O3 -m64 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex --enable-thread-safe-client
--enable-local-infile --enable-assembler --with-named-z-libs=no
--with-named-curses-libs=-lcurses --disable-shared
```

Sun Solaris 9 sparc com gcc 2.95.3

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3
-fno-omit-frame-pointer -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--enable-assembler --with-named-curses-libs=-lcurses --disable-
shared
```

Sun Solaris 9 sparc com `cc-5.0` (Sun Forte 5.0)

```
CC=cc-5.0 CXX=CC ASFLAGS="-xarch=v9" CFLAGS="-Xa -xstrconst
-mt -D_FORTEC_ -xarch=v9" CXXFLAGS="-noex -mt -D_FORTEC_
-xarch=v9" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--enable-assembler --with-named-z-libs=no --enable-thread-safe-
client --disable-shared
```

IBM AIX 4.3.2 ppc com `gcc` 3.2.3

```
CFLAGS="-O2 -mcpu=powerpc -Wa,-many " CXX=gcc CXXFLAGS="-O2
-mcpu=powerpc -Wa,-many -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--with-named-z-libs=no --disable-shared
```

IBM AIX 4.3.3 ppc com `xlC_r` (IBM Visual Age C/C++ 6.0)

```
CC=xlc_r CFLAGS="-ma -O2 -qstrict -qoptimize=2 -qmaxmem=8192"
CXX=xlC_r CXXFLAGS ="-ma -O2 -qstrict -qoptimize=2 -qmaxmem=8192"
./configure --prefix=/usr/local/mysql --localstatedir=/usr/local/mysql/data
--libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --with-named-z-
libs=no --disable-shared --with-innodb
```

IBM AIX 5.1.0 ppc com `gcc` 3.3

```
CFLAGS="-O2 -mcpu=powerpc -Wa,-many" CXX=gcc CXXFLAGS="-O2
-mcpu=powerpc -Wa,-many -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --with-server-suffix="-pro" --enable-thread-
safe-client --enable-local-infile --with-named-z-libs=no
--disable-shared
```

HP-UX 10.20 pa-risc1.1 com `gcc` 3.1

```
CFLAGS="-DHPUX -I/opt/dce/include -O3 -fPIC" CXX=gcc CXXFLAGS="-
DHPUX -I/opt/dce /include -felide-constructors -fno-exceptions
-fno-rtti -O3 -fPIC" ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex --enable-thread-safe-client --
enable-local-infile --with-pthread --with-named-thread-libs=-ldce
--with-lib-ccflags=-fPIC --disable-shared
```

HP-UX 11.11 pa-risc2.0 64 bit com `aCC` (HP ANSI C++ B3910B A.03.33)

```
CC=cc CXX=aCC CFLAGS=+DD64 CXXFLAGS=+DD64 ./configure --
prefix=/usr/local/mysql --with-extra-charsets=complex --enable-
thread-safe-client --enable-local-infile --disable-shared
```

HP-UX 11.11 pa-risc2.0 32bit com `aCC` (HP ANSI C++ B3910B A.03.33)

```
CC=cc CXX=aCC CFLAGS="+DAportable" CXXFLAGS="+DAportable"
./configure --prefix=/usr/local/mysql --localstatedir=/usr/local/mysql/data
--libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared
--with-innodb
```

Apple Mac OS X 10.2 powerpc com gcc 3.1

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3
-fno-omit-frame-pointer -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--disable-shared
```

FreeBSD 4.7 i386 com gcc 2.95.4

```
CFLAGS=-DHAVE_BROKEN_REALPATH ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex --enable-thread-safe-client
--enable-local-infile --enable-assembler --with-named-z-libs=not-
used --disable-shared
```

QNX Neutrino 6.2.1 i386 with gcc 2.95.3qnx-nto 20010315

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3
-fno-omit-frame-pointer -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--disable-shared
```

Os seguintes binários são contruídos em sistemas de terceiros gentilmente cedidos para a MySQL AB pou outros usuários. Pou favor, note que eles só são fornecidos como cortesia. Uma vez que a MySQL AB não tem total controle sobre estes sistemas, nós podemos fornecer apenas suporte limitado para os binários construídos nestes sistemas.

SCO Unix 3.2v5.0.6 i386 com gcc 2.95.3

```
CFLAGS="-O3 -mpentium" LDFLAGS=-static CXX=gcc CXXFLAGS="-O3 -
mpentium -felide-constructors" ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex --enable-thread-safe-client
--enable-local-infile --with-named-z-libs=no --enable-thread-safe-
client --disable-shared
```

SCO OpenUnix 8.0.0 i386 com CC 3.2

```
CC=cc CFLAGS="-O" CXX=CC ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex --enable-thread-safe-client
--enable-local-infile --with-named-z-libs=no --enable-thread-safe-
client --disable-shared
```

Compaq Tru64 OSF/1 V5.1 732 alpha com cc/cxx (Compaq C V6.3-029i / DIGITAL C++ V6.1-027)

```
CC="cc -pthread" CFLAGS="-O4 -ansi_alias -ansi_args -fast -
inline speed -speculate all" CXX="cxx -pthread" CXXFLAGS="-O4
-ansi_alias -fast -inline speed -speculate all -noexceptions
-nortti" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--with-prefix=/usr/local/mysql --with-named-thread-libs="-
lpthread -lmach -lexc -lc" --disable-shared --with-mysqld-ldflags=-
all-static
```

SGI Irix 6.5 IP32 com gcc 3.0.1

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXXFLAGS="-O3
-fno-omit-frame-pointer -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--disable-shared
```

FreeBSD 5.0 sparc64 com gcc 3.2.1

```
CFLAGS=-DHAVE_BROKEN_REALPATH ./configure --prefix=/usr/local/mysql
--localstatedir=/usr/local/mysql/data --libexecdir=/usr/local/mysql/bin
--with-extra-charsets=complex --enable-thread-safe-client
--enable-local-infile --disable-shared --with-innodb
```

As seguintes opções de compilação foram usadas nos pacotes binários que a MySQL AB costumava fornecer no passado. Estes binários não são mais atualizados, mas as opções de compilação são mantidas aqui com o propósito de referência.

Linux 2.2.xx sparc com egcs 1.1.2

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3
-fno-omit-frame-pointer -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-
charsets=complex --enable-thread-safe-client --enable-local-infile
--enable-assembler --disable-shared
```

Linux 2.2.x com x686 com gcc 2.95.2

```
CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro
-felide-constructors -fno-exceptions -fno-rtti" ./configure
--prefix=/usr/local/mysql --enable-assembler --with-mysqld-
ldflags=-all-static --disable-shared --with-extra-charsets=complex
```

SunOS 4.1.4 2 sun4c com gcc 2.7.2.1

```
CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors" ./configure
--prefix=/usr/local/mysql --disable-shared --with-extra-
charsets=complex --enable-assembler
```

SunOS 5.5.1 (e acima) sun4u com egcs 1.0.3a ou 2.90.27 ou gcc 2.95.2 e mais novo

```
CC=gcc CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql
--with-low-memory --with-extra-charsets=complex --enable-assembler
```

SunOS 5.6 i86pc com gcc 2.8.1

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
--with-low-memory --with-extra-charsets=complex
```

BSDI BSD/OS 3.1 i386 com gcc 2.7.2.1

```
CC=gcc CXX=gcc CXXFLAGS=-O ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex
```

BSDI BSD/OS 2.1 i386 com gcc 2.7.2

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex
```

AIX 2 4 com `gcc` 2.7.2.2

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex
```

Qualquer que tenha mais opções otimizadas para qualquer das configurações listadas acima pode sempre enviá-los para a lista de email "internals" do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33.

Distribuições RPM que anteceda o MySQL versão 3.22 são contribuições dos usuários. Os RPMs gerados por nós da MySQL AB só começaram a ser fornecidos a partir da versão 3.22 do MySQL.

Se você deseja compilar uma versão para depuração do MySQL, você deve adicionar `--with-debug` ou `--with-debug=full` para as linhas de configuração acima e remover qualquer opção `-fomit-frame-pointer`.

Para distribuições do Windows, por favor, veja Seção 2.1.1 [Windows installation], Página 60.

2.2.9 Instalando uma Distribuição Binária do MySQL

Este capítulo cobre a instalação da distribuição binária do MySQL (`.tar.gz` Archives) para várias plataformas (veja `MySQL binaries` para uma lista detalhada).

Em adição a estes pacotes genéricos, também oferecemos binários em formatos de pacote específicos da plataforma para plataformas selecionadas. Veja Seção 2.1 [Quick Standard Installation], Página 60 para mais informações sobre como\ intalá-los.

As distribuições genéricas do MySQL estão empacotados com arquivos GNU tar com compactação gzip (`.tar.gz`). Você precisa das seguintes ferramentas para instalar um distribuição binária do MySQL:

- GNU `gunzip` para descompactar a distribuição.
- Um `tar` razoável para descompactar a distribuição. Sabemos que o GNU `tar` funciona. Algumas implementações `tar` que vêm pré-instaladas como o sistema operacional (ex. Sun `tar`) possuem problemas (com nome de arquivos grandes, por exemplo) Neste caso, você deve instalar o GNU `tar` primeiro.

Se você tiver problemas, **sempre use** `mysqlbug` ao enviar dúvidas para a lista de email do MySQL. Mesmo se o problema não for um bug, `mysqlbug` colhe informações do sistema que ajudarão os outros a solucionar o seu problema. Se não usar `mysqlbug`, você terá diminuída a probabilidade de conseguir a solução do seu problema. Você encontrará o `mysqlbug` no diretório '`bin`' depois de descompactar a distribuição. Veja Seção 1.7.1.3 [Relato de erros], Página 36.

Os comando básicos que você deve executar para instalar e usar uma distribuição binária do MySQL são:

```
shell> groupadd mysql
shell> useradd -g mysql mysql
shell> cd /usr/local
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
```

```
shell> scripts/mysql_install_db
shell> chown -R root  .
shell> chown -R mysql data
shell> chgrp -R mysql .
shell> bin/mysqld_safe --user=mysql &
```

Se a sua versão do MySQL é mais antiga que a 4.0, substitua `bin/safe_mysqld` por `bin/mysqld_safe` no comando final.

Você pode adicionar novos usuários usando o script `bin/mysql_setpermission` se você instalar os módulos Perl `DBI` e `DBD-mysql`.

Uma descrição mais detalhada é apresentada a seguir.

Para instalar uma distribuição binária, siga estes passos, então proceda com a Seção 2.4 [Pós Instalação], Página 111, para a configuração da pós istalação e testes:

1. Escolha o diretório sob o qual você deseja descompactar a distribuição e a mova para dentro dele. No exemplo a seguir, descompactamos a distribuição sob '`/usr/local`' e criamos um diretório '`/usr/local/mysql`' dentro do qual o MySQL é instalado. (As seguintes instruções, consequentemente, assumem que você tem permissão para criar arquivos em '`/usr/local`'. Se este diretório é protegido, você precisará realizar a instalação como `root`.)
2. Obtenha um arquivo de distribuição de um dos sites listados em Seção 2.2.1 [Getting MySQL], Página 75.

   As distribuições binárias do MySQL são fornecidas como arquivos `tar` compactados e tem nomes como '`mysql-VERSÃO-SO.tar.gz`', onde `VERSÃO` é um número (por exemplo, `3.21.15`) e `SO` indica o tipo de sistema operacional para o qual a distribuição é pretendida (por exemplo, `pc-linux-gnu-i586`).
3. Se você ver uma distribuição binária marcada com o sufixo `-max`, significa que o binário tem suporte para tabelas transacionais e outros recursos. Veja Seção 4.8.5 [`mysqld-max`], Página 343. Note que todos os binários são contruídos a partir da mesma distribuição fonte do MySQL.
4. Adicione um usuário e grupo para o `mysqld` ser executado:

   ```
   shell> groupadd mysql
   shell> useradd -g mysql mysql
   ```

   Estes comandos adicionam o grupo `mysql` e o usuário `mysql`. A sintaxe para `useradd` e `groupadd` podem diferir um pouco nas diversas versões de Unix. Eles tambémpodem ser chamado `adduser` e `addgroup`. Você pode desejar criar o grupo e usuário com outro nome, diferente de `mysql`.
5. Chame o diretório no qual se pretende fazer a instalação:

   ```
   shell> cd /usr/local
   ```
6. Descompacte a distribuição, que criará o diretório de instalação. Então crie um link simbólico para aquele diretório:

   ```
   shell> gunzip < /path/to/mysql-VERSÃO-SO.tar.gz | tar xvf -
   shell> ln -s full-path-to-mysql-VERSÃO-SO mysql
   ```

   O primeiro comando cria um diretório chamado '`mysql-VERSÃO-SO`'. O segundo comando cria um link simbólico para o diretório. Isto torna a referência ao diretório de instalação mais fácil, chamado como '`/usr/local/mysql`'.

7. Altere para p diretório de instalação:

   ```
   shell> cd mysql
   ```

   Você encontrará diversos arquivos e subdiretórios no diretório `mysql`. O mais importante para propósitos de instalação são os subdiretórios '`bin`' e '`scripts`'.

   '`bin`' Este diretório contém o programa cliente e o servidor. Você deve adicionar o caminho completo deste diretório a sua variável de ambiente `PATH` e assim a sua shell encontrará o programa MySQL de forma apropriada. Veja Apêndice F [Variáveis de ambiente], Página 1092.

   '`scripts`' Este diretório contém o script `mysql_install_db` usado para inicializar o banco de dados `mysql` contendo a tabela de permissões que armazenam o servidor de permissões de acesso.

8. Caso você desejasse usar o `mysqlaccess` e a distribuição do MySQL está em um local diferente do padrão, você deve alterar a localização para onde o `mysqlaccess` espera encontrar o cliente `mysql`. Edite o script '`bin/mysqlaccess`' aproximadamente na linha 18. Procure pela linha que se parece com a apresentada abaixo:

   ```
   $MYSQL     = '/usr/local/bin/mysql';    # path to mysql executable
   ```

   Altere o caminho para o local onde o `mysql` atualmente está armazaenado em seu sistema. Se você não fizer isto receberá uma mensagem de erro `Broken pipe` quando executar o `mysqlaccess`.

9. Crie as tabelas de permissão do MySQL (necessário apenas se você não tiver instalado o MySQL anteriormente):

   ```
   shell> scripts/mysql_install_db
   ```

   Note que as versões mais antigas que a 3.22.10 iniciam o servidor MySQL quando você executa o `mysql_install_db`. Isto não ocorre mais.

10. Altere o proprietário dos binários para o `root` e o proprietário do diretório de dados para o usuário com o quel você executará o `mysqld`:

    ```
    shell> chown -R root  /usr/local/mysql/.
    shell> chown -R mysql /usr/local/mysql/data
    shell> chgrp -R mysql /usr/local/mysql/.
    ```

    O primeiro comando altera o atributo `owner` dos arquivos para o usuário `root`, o segundo altera o atributo `owner` do diretório de dados para o usuário `mysql` e o terceiro altera o atributo `group` para o grupo `mysql`.

11. Se você quiser instalar o suporte para a interface Perl `DBI/DBD`, veja Seção 2.7 [Perl support], Página 164.

12. Se você desejasse que o MySQL seja iniciado automaticamente quando você iniciar a sua máquina, você pode copiar `support-files/mysql.server` para o local onde o seu sistema tem os arquivos de inicialização. Mais informações podem ser encontradas no script `support-files/mysql.server` e em Seção 2.4.3 [Automatic start], Página 118.

Depois de tudo estar descompactado e instalado, você deve inicializar e testar a sua distribuição.

Você pode iniciar o servidor MySQL com o seguinte comando:

```
shell> bin/mysqld_safe --user=mysql &
```

Se a sua versão do MySQl for mais antiga do que a 4.0, substitua `bin/safe_mysqld` por `bin/mysqld_safe` no comando.

Agora prossiga com Seção 4.8.2 [`mysqld_safe`], Página 331 e Veja Seção 2.4 [Pós instalação], Página 111.

2.3 Instalando uma distribuição com fontes do MySQL

Antes de você continuar com as instalações dos fontes, confira antes se nosso binário está disponível para sua plataforma e se ela funcionará para você. Nós colocamos muito esforço para ter certeza que nossos binários são contruídos com as melhores opções possíveis.

Você precisa das seguintes ferramentas para contruir e instalar o MySQL a partir do código fonte:

- GNU `gunzip` para descompactar a distribuição.
- Um `tar` razoável para desempacotar a distribuição. Sabe-se que o GNU `tar` funciona. Algumas implementações `tar` que vêm pré-instaladas como o sistema operacional (ex. Sun `tar`) possuem problemas (com nome de arquivos grandes, por exemplo) Neste caso, você deve instalar o GNU `tar` primeiro.
- Um compilador ANSI C++ funcional. `gcc` >= 2.95.2, `egcs` >= 1.0.2 ou `egcs 2.91.66`, SGI C++, e SunPro C++ são alguns dos compiladores que sabemos que funcionam. A `libg++` não é necessária quando o `gcc` for usado. `gcc` 2.7.x tem um bug que torna impossível compilar alguns arquivos C++ perfeitamente corretos, como o '`sql/sql_base.cc`'. Se você possui somente o `gcc` 2.7.x você deve atualiza-lo para conseguir compilar o MySQL. `gcc` 2.8.1 é também conhecido por ter problemas em algumas plataformas portanto ele deve ser evitado se existir um novo compilador para a plataforma.

  `gcc` >= 2.95.2 é recomendado quando compilar o MySQL Versão 3.23.x.
- Um bom programa `make`. GNU `make` é sempre recomendado e é algumas vezes necessário. Se você tiver problemas, recomendamos tentar o GNU `make` 3.75 ou mais novo.

Se você estiver usando uma versão recente de **gcc**, recente o bastante para entender a opção `-fno-exceptions`, é **MUITO IMPORTANTE** que você a use. De outra forma, você pode compilar um binário que quebra randomicamente. Nós também recomendamos que você use `-felide-constructors` e `-fno-rtti` juntas com `-fno-exception`. Se estiver com dúvidas, faça o seguinte:

```
CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions \
       -fno-rtti" ./configure --prefix=/usr/local/mysql --enable-assembler \
       --with-mysqld-ldflags=-all-static
```

Na maioria dos sistemas você irá obter um binário rápido e estável com essas opções.

Se você tiver problemas, **SEMPRE USE** `mysqlbug` quando postar questões para a lista de email do MySQL Mesmo se o problema não for um bug, `mysqlbug` recolhe informações do sistema que facilitará aos outros resolverem seu problema. Por não suar `mysqlbug`, você perde a vantagem de ter seu problema resolvido! Você irá encontrar `mysqlbug` no

diretório '`scripts`' depois de desempacotar a distribuição. Veja Seção 1.7.1.3 [Bug reports], Página 36.

2.3.1 Visão geral da instalação rápida

Os comandos básicos que você deve executar para instalar o MysQL a partir da distribuição fonte são:

```
shell> groupadd mysql
shell> useradd -g mysql mysql
shell> gunzip < mysql-VERSION.tar.gz | tar -xvf -
shell> cd mysql-VERSION
shell> ./configure --prefix=/usr/local/mysql
shell> make
shell> make install
shell> scripts/mysql_install_db
shell> chown -R root  /usr/local/mysql
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

Se a sua versão do MySQL é mais antiga que a 4.0, substitua `bin/safe_mysqld` por `bin/mysqld_safe` no comando final.

Se você deseja ter suporte para tabelas InnoDB, você deve editar o arquivo `/etc/my.cnf` e remover o caractere `#` antes dos parâmetros que iniciam com `innodb_...`. Veja Seção 4.1.2 [Option files], Página 216. Veja Seção 7.5.3 [InnoDB start], Página 644.

Se você iniciar de um RPM fonte, então faça o seguinte:

```
shell> rpm --rebuild --clean MySQL-VERSION.src.rpm
```

Isto irá criar um RPM binário que você pode instalar.

Você pode adicionar novos usuários utilizando o script `bin/mysql_setpermission` se você instalar os módulos Perl `DBI` e `DBD-mysql`.

Segue uma descrição mais detalhada.

Para instalar uma distribuição fonte, siga os passos a seguir, então prossiga para Seção 2.4 [Post-installation], Página 111, para inicialização do pós-instalação e testes:

1. Escolha o diretório sobre o qual você deseja descompactar a distribuição e vá para ele.
2. Obtenha um arquivo de distribuição de algum dos sites listados em Seção 2.2.1 [Getting MySQL], Página 75.
3. Se você esta interessado em usar tabelas Berkeley DB com MySQL, você precisará obter uma versão com o patch do código fonte do Berkeley DB. Por favor leia o capítulo sobre tabelas Berkeley DB antes de continuar. Veja Seção 7.6 [BDB], Página 698.

   Distribuições fontes do MySQL são fornecidas como arquivos `tar` compactados e tem nomes como '`mysql-VERSION.tar.gz`', onde `VERSION` é um número como 5.0.0-alpha.
4. Adicione um usuário e grupo para o `mysql` executar assim:

```
shell> groupadd mysql
shell> useradd -g mysql mysql
```

Estes comandos adicionam o grupo `mysql` e o usuário `mysql`. A sintaxe para `useradd` e `groupadd` podem mudar um pouco em diferentes versões de Unix. Elas podem também ser chamadas `adduser` e `addgroup`. Você pode escolher outros nomes para o usuário e grupo em vez de `mysql`.

5. Descompacte a distribuição para o diretório corrente:

   ```
   shell> gunzip < /path/to/mysql-VERSION.tar.gz | tar xvf -
   ```

   Este comando cria um diretório com o nome '`mysql-VERSION`'.

6. Mude para o diretório da distribuição descompactada:

   ```
   shell> cd mysql-VERSION
   ```

   Note que agora você deve configurar e construir o MySQL a partir deste diretório raiz da distribuição. Você não pode construí-lo em um diretório diferente.

7. Configure o release e compile tudo:

   ```
   shell> ./configure --prefix=/usr/local/mysql
   shell> make
   ```

   Quando você executar `configure`, você pode desejar especificar algumas opções. Execute `./configure --help` para uma lista das opções. Seção 2.3.3 [`configure` options], Página 98, discute algumas das opções mais usadas.

   Se o `configure` falhar, e você for enviar uma mensagem para lista de email do MySQL para pedir ajuda, por favor, inclua qualquer linhas de '`config.log`' que você acha que pode ajudar a resolver o problema. Também inclua as últimas linhas da saída de `configure` se o `configure` abortar. Envie o relatório de erros usando o script `mysqlbug`. Veja Seção 1.7.1.3 [Bug reports], Página 36.

   Se a compilação falhar, veja Seção 2.3.5 [Compilation problems], Página 103, para uma ajuda com um varios problemas comuns.

8. Instalar tudo:

   ```
   shell> make install
   ```

   Você deve executar este comando como `root`.

9. Crie as tabelas de permissões do MySQL (necessárias só se você não tiver instalado o MySQL anteriormente):

   ```
   shell> scripts/mysql_install_db
   ```

   Note que as versões do MySQL anteriores à versão 3.22.10 iniciam o servidor MySQL quando você executa `mysql_install_db`. Isto não acontece mais!

10. Altere o dono dos binários para `root` e do diretório dados para o usuário que irá executar o `mysqld`:

    ```
    shell> chown -R root  /usr/local/mysql
    shell> chown -R mysql /usr/local/mysql/var
    shell> chgrp -R mysql /usr/local/mysql
    ```

    O primeiro comando altera o atributo de `proriedade` dos arquivos para o usuário `root`, o segundo altera o atributo de `propriedade` do diretório de dados para o usuário `mysql`, e o terceiro altera o atributo de `grupo` para o grupo `mysql`.

11. Se você deseja instalar suporte para a interface Perl `DBI/DBD`, veja Seção 2.7 [Perl support], Página 164.
12. Se você deseja que o MySQL inicie automaticamente quando você ligar sua máquina, você pode copiar `support-files/mysql.server` para o local onde seu sistema tem seus arquivos de incialização. Mais informações podem ser encontradas no próprio script `support-files/mysql.server` e em Seção 2.4.3 [Automatic start], Página 118.

Depois de tudo ter sido instalado, você deve iniciar e testar sua distribuição usando este comando:

```
shell> /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

Se a sua versão do MySQL for mais antiga do que 4.0, substitua `safe_mysqld` por `mysqld_safe` no comando:

Se o comando falhar imediatamente com `mysqld daemon ended` então você pode achar alguma informação no arquivo '`diretório-dados-mysql/'nome_maquina'.err`'. A razão pode ser que você já possua outro servidor `mysqld` sendo executado. Veja Seção 4.2 [Multiple servers], Página 219.

Veja Seção 2.4 [Post-installation], Página 111.

2.3.2 Aplicando patches

Algumas vezes patches aparecem na lista de mensagens ou são colocados na área de patches do MySQL. (`http://www.mysql.com/downloads/patches.html`).

Para aplicar um patch da lista de mensagens, salve a mensagem em que o patch aparece em um arquivo, mude para o diretório raiz da sua distribuição fonte de seu MySQL e execute estes comandos:

```
shell> patch -p1 < patch-file-name
shell> rm config.cache
shell> make clean
```

Patches do site FTP são distribuídos como arquivos texto ou como arquivos compactados com `gzip`. Aplique um patch no formato texto como mostrado acima para patches da lista de mensagens. Para aplicar um patch compactado, mude para o diretório raiz da árvore fonte do MySQL e execute estes comandos:

```
shell> gunzip < patch-file-name.gz | patch -p1
shell> rm config.cache
shell> make clean
```

Depois de aplicar um patch siga as instruções para uma instalação normal a partir dos fontes começando com o passo `./configure`. Depois de executar o passo `make install`, reinicie seu servidor MySQL.

Você pode precisar derrubar algum servidor atualmente em execução antes de executar `make install`. (Use `mysqladmin shutdown` para fazer isto.) Alguns sistemas não lhe permitem instalar uma nova versão do programa se ele substitui agum que estiver em execução.

2.3.3 Opções típicas do `configure`

O script `configure` fornece uma grande gama de controle sobre como você configura sua distribuição MySQL. Normalmente você faz isto usando opções na linha de comando do `configure`. Você também pode alterar `configure` usando algumas variáveis de ambiente. Veja Apêndice F [Environment variables], Página 1092. Para uma lista de opções suportadas pelo `configure`, execute este comando:

```
shell> ./configure --help
```

Algumas das opções mais usadas normalmente com o `configure` estão descritas a seguir:

- Para compilar apenas as bibliotecas clientes do MySQL e programas clientes e não o servidor, use a opção `--without-server`:

  ```
  shell> ./configure --without-server
  ```

  Se você não possui um compilador C++, `mysql` não irá compilar (ele é o programa cliente que exige C++). Neste caso, você pode remover o código no `configure` que testa pelo compilador C++ e executar `./configure` com a opção `--without-server`. O passo da compiação continuará tentaindo construir `mysql`, mas você pode ignorar as advertências sobre '`mysql.cc`'. (Se o `make` parar, tente `make -k` para continuar com o resto da compilação mesmo se erros ocorrerem.)
- Se você quiser uma biblioteca embutida do MySQL (`libmysqld.a`) você deve usar a opção `--with-embedded-server`.
- Se você não deseja que seus arquivos de logs e diretórios de bancos de dados fiquem localizados sobre '`/usr/local/var`', use o comando `configure`; algo parecido com um destes:

  ```
  shell> ./configure --prefix=/usr/local/mysql
  shell> ./configure --prefix=/usr/local \
             --localstatedir=/usr/local/mysql/data
  ```

  O primeiro comando altera o diretório instalação para que tudo seja instalado sobre '`/usr/local/mysql`' em vez do padrão '`/usr/local`'. O segundo comando preserva o diretório da instalação padrão, mas altera a localização padrão para diretórios de bancos de dados (normalmente '`/usr/local/var`') e altera para `/usr/local/mysql/data`. Depois de ter compilado o MySQL, você pode alterar estas opçãoes com arquivos de opções. Veja Seção 4.1.2 [Option files], Página 216.
- Se você estiver usando Unix e deseja que o arquivo socket do MySQL fique em um diretório diferente do padrão (normalmente no diretório '`/tmp`' ou '`/var/run`') use o comando `configure` da seguinte forma:

  ```
  shell> ./configure --with-unix-socket-path=/usr/local/mysql/tmp/mysql.sock
  ```

  Perceba que o arquivo fornecido deve ter um caminho absoluto ! Você também pode, mais tarde, alterar a localização de '`mysql.sock`' usando os arquivos de opções do MySQL. Veja Seção A.4.5 [Problems with mysql.sock], Página 929.
- Se você deseja compilar programas linkeditados estaticamente (por exemplo, para criar uma distribuição binária, obter mais velocidade, ou evitar problemas com algumas distribuições Red Hat Linux), execute `configure` desta forma:

  ```
  shell> ./configure --with-client-ldflags=-all-static \
             --with-mysqld-ldflags=-all-static
  ```

- Se você estiver usando `gcc` e não tem `libg++` ou `libstdc++` instalados você pode dizer ao `configure` para usar o `gcc` como seu compilador C++:

  ```
  shell> CC=gcc CXX=gcc ./configure
  ```

  Quando você usar `como` seu compilador C++, ele não tentará ligar com o `libg++` ou `libstdc++`. Isto pode ser uma boa idéia para se fazer se você tiver as bibliotecas acimas instaladas, já que algumas versões destas bibliotecas tem causado problemas estranhos para usuários do MySQL no passado.

  Segue algumas configurações de variáveis de ambiente comuns, dependendo do compilador que você estiver usando:

Compiler	**Recommended options**
gcc 2.7.2.1	CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors"
egcs 1.0.3a	CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti"
gcc 2.95.2	CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro \ -felide-constructors -fno-exceptions -fno-rtti"
pgcc 2.90.29 or newer	CFLAGS="-O3 -mpentiumpro -mstack-align-double" CXX=gcc \ CXXFLAGS="-O3 -mpentiumpro -mstack-align-double -felide-constructors \ -fno-exceptions -fno-rtti"

  Na maioria dos casos você pode obter um binário MySQL razoavelmente otimizado usando as opções acima e adicionar as seguintes opções para a linha de configuração:

  ```
  --prefix=/usr/local/mysql --enable-assembler \
  --with-mysqld-ldflags=-all-static
  ```

  A linha completa de configuração deverá ser, em outras palavras, algo como o seguinte para todas as versões recentes do gcc:

  ```
  CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro \
  -felide-constructors -fno-exceptions -fno-rtti" ./configure \
  --prefix=/usr/local/mysql --enable-assembler \
  --with-mysqld-ldflags=-all-static
  ```

  Os binários que fornecemos no site Web MySQL em `http://www.mysql.com` são todos compilados com otimização plena e deve ser perfeito para a maioria dos usuários. Veja Seção 2.2.8 [Binários do MySQL], Página 86. Existem algumas definições de configuração que você pode alterar para criar um binário ainda mais rápido, mas isto é somente para usuários avançados. Veja Seção 5.5.3 [Compile and link options], Página 456.

  Se a construção falhar e produzir erros sobre seu compilador ou linkeditor não estarem aptos para criarem a biblioteca compartilhada '`libmysqlclient.so.r#`' ('`r#`' é um número de versão), você pode evitar este problema fornecendo a opção `--disable-share` para o `configure`. Neste caso, `configure` não construirá uma biblioteca `libmysqlclient.so.*` compartilhada.

- Você pode configurar o MySQL para não usar valores de campos `DEFAULT` para campos não-`NULL` (isto é, campos que não podem ser `NULL`). Veja Seção 1.8.5.2 [Restrições NOT NULL], Página 53.

  ```
  shell> CXXFLAGS=-DDONT_USE_DEFAULT_FIELDS ./configure
  ```

- Por padrão, o MySQL usa o conjunto de caracteres ISO-8859-1 (Latin1). Para alterar o conjunto padrão, use a opção `--with-charset`:

```
shell> ./configure --with-charset=CHARSET
```

`CHARSET` pode ser um de `big5`, `cp1251`, `cp1257`, `czech`, `danish`, `dec8`, `dos`, `euc_kr`, `gb2312`, `gbk`, `german1`, `hebrew`, `hp8`, `hungarian`, `koi8_ru`, `koi8_ukr`, `latin1`, `latin2`, `sjis`, `swe7`, `tis620`, `ujis`, `usa7`, ou `win1251ukr`. Veja Seção 4.7.1 [Conjunto de caracteres], Página 325.

Se você desja converter os caracteres entre o servidor e o cliente, você deve dar uma olhada no comando `SET OPTION CHARACTER SET`. Veja Seção 5.5.6 [`SET OPTION`], Página 460.

Cuidado: Se você alterar o conjunto de caracteres depois de ter criado qualquer tabela, você deve executar `myisamchk -r -q --set-character--set=charset` em cada tabela. Seus índices podem ser ordenados incorretamente. (Isto pode acontecer se você instalar o MySQL, criar algumas tabelas, depois reconfigurar o MySQL para usar um conjunto diferente de caracteres e reinstalá-lo).

Com a opção `--with-extra-charset=LISTA` você pode definir qual conjunto de caracteres adicionais deve ser compilado no servidor.

Aqui `LISTA` é uma lista de conjuntos de caracteres separados por espaços, `complex` para incluir todos caracteres que não podem ser carregados dinamicamente ou `all` para incluir todos os conjuntos nos binários.

- Para configurar o MySQL com código para depuração, use a opção `--with-debug`:

  ```
  shell> ./configure --with-debug
  ```

  Isto inclui uma alocação segura de memória que pode encontrar alguns erros e fornecer saída sobre o que está acontecendo. Veja Seção E.1 [Depurando o servidor], Página 1079.
- Se seus programas clientes usam threads, você precisará também compilar uma versão thread-safe da biblioteca cliente do MySQL com as opções do configure `--enable-thread-safe-client`. Isto irá criar uma biblioteca `libmysqlclient_r` com o qual você deverá ligar suas aplicações que fazem uso de threads. Veja Seção 12.1.14 [Clientes em threads], Página 863.
- Opções que pertençam a sistemas particulares podem ser encontrados na seção com detalhes específicos de sistemas neste manual. Veja Seção 2.6 [Notas específicas do Sistema Operacional], Página 133.

2.3.4 Instalando pela árvore de fontes do desenvolvimento

CUIDADO: Você deve ler esta seção somente se você estiver interessado em nos ajudar a testar nossos novos códigos. Se você só deseja deixar o MySQL funcionando em seus sistema, você deve usar uma distribuição padrão (pode ser uma distribuição binária ou fonte).

Para obter noss mais nova árvore de desenvolvimento, use estas instruções:

1. Faça download do **BitKeeper** em `http://www.bitmover.com/cgi-bin/download.cgi`. Você precisará do **Bitkeeper** 3.0 ou posterior para acessar nosso repositório.
2. Siga as instruções para instalá-lo.
3. Depois que o **BitKeeper** estiver instalado, primeiro vá ao diretório no qual você deseja trabalhar e então use um dos seguintes comandos para clonar o ramo da versão MySQL de sua escolha:

Para clonar o ramo 3.23 (antigo), use este comando:

```
shell> bk clone bk://mysql.bkbits.net/mysql-3.23 mysql-3.23
```

Para clonar o ramo 4.0 (estável/produção), use este comando:

```
shell> bk clone bk://mysql.bkbits.net/mysql-4.0 mysql-4.0
```

Para clonar o ramo 4.1 alfa, use este comando:

```
shell> bk clone bk://mysql.bkbits.net/mysql-4.1 mysql-4.1
```

Para clonar o ramo de desenvolvimento 5.0, use este comando:

```
shell> bk clone bk://mysql.bkbits.net/mysql-5.0 mysql-5.0
```

Nos exemplos anteriores a árvore binária será configurada no subdiretório '`mysql-3.23/`', '`mysql-4.0/`', '`mysql-4.1/`', ou '`mysql-5.0/`' do diretório atual.

Se você está atrás de um firewall e só pode iniciar conexões HTTP, você também pode o `BitKeeper` via HTTP.

Se vocÊ precisa usar um servidor proxy, simplesmente configure a variável de ambiente `http_proxy` para apontar para o seu proxy:

```
shell> export http_proxy="http://seu.servidor.proxy:8080/"
```

Agora, simplesmente substitua o `bk://` com o `http://` ao fazer um clone. Exemplo:

```
shell> bk clone http://mysql.bkbits.net/mysql-4.1 mysql-4.1
```

O download inicial da árvore fonte pode demorar um pouco, dependendo da velocidade de sua conexão; seja paciente.

4. Você precisará do **GNU** `make`, `autoconf 2.53 (ou posterior)`, `automake 1.5`, `libtool 1.4` e `m4` para executar o próximo conjunto de comandos. Embora muitos sistemas operacionais já venham com suas próprias implementações do `make`, as chances de que a sua compilação falhe com mensagens de erros estranhas são altas. Consequentemente é altamente recomendado usar o GNU `make` (algumas vezes também chamado `gmake`).

   Felizmente, um grande número de sistemas operacionais já vem com a ferramente GNU pré instalada ou são fornecidos pacotes de instalação da mesma. De qualquer forma, elas podem ser encontradas nos seguintes locais:

   - `http://www.gnu.org/software/autoconf/`
   - `http://www.gnu.org/software/automake/`
   - `http://www.gnu.org/software/libtool/`
   - `http://www.gnu.org/software/make/`

   Se você estiver tentando configurar o MySQL 4.1 você também precisará do `bison 1.75`. Versões mais antigas do `bison` podem exiobir este erro: `sql_yacc.yy:#####: fatal error: maximum table size (32767) exceeded`. Nota: o tamanho máximo da tabela não é realmente excedido, o erro é causado por um bug nas versões mais novas do `bison`.

   Versões do MySQL anteriores a 4.1 podem também compilar com outras implementações `yacc` (e.g. BSD `yacc` 91.7.30). Para versões posteriores, GNU `bison` é uma exigência.

   O comando comum para fazer em uma shell é:

```
cd mysql-4.0
bk -r edit
aclocal; autoheader; autoconf; automake
(cd innobase; aclocal; autoheader; autoconf; automake) # for InnoDB
(cd bdb/dist; sh s_all ) # for Berkeley DB
./configure  # Adicione suas opções favoritas aqui
make
```

Caso apareçam alguns erros estranhos durantes este estágio, confira se você realmente tem a `libtool` instalada!

Uma coleção de nossos scripts de configuração padrões está localizada no subdiretório '`BUILD/`'. Se preferir, você pode usar '`BUILD/compile-pentium-debug`'. Para compilar em uma arquitetura diferente, modifique o script removendo opções que são específicas da arquitetura Pentium.

5. Quando a construção estiver pronta, execute `make install`. Seja cuidadoso com isto em uma máquina de produção; o comando pode sobrescrever sua versão atual instalada. Se você tem outra instalação do MySQL, nós recomendamos que você execute `./configure` com valores diferentes para as opções `prefix`, `tcp-port` e `unix-socket-path` que as usadas pelo seu servidor em produção.
6. Seja rígido com sua nova instalação e tente fazer com que os novos recursos falhem. Inicie executando `make test`. Veja Seção 14.1.2 [MySQL test suite], Página 896.
7. Se você chegar ao estágio `make` e a distribuição não compilar, por favor relate-o para `bugs@lists.mysql.com`. Se você instalou as últimas versões das ferramentas GNU exigidas, e elas falharam tentando processar nossos arquivos de configuração, por favor informe isto também. Entretanto, se você executar `aclocal` e obtêm um erro de `command not found` não o reporte.Tenha certeza que todas as ferramentas necessárias estejam instaladas e que sua variável `PATH` esteja corretamente configurada para que sua shell possa encontrá-la.
8. Depois da operação inicial `bk clone` para obter a árvore fonte, você deve executar `bk pull` periodicamente para obter as atualizações.
9. Você pode examinar o histórico de alterações para a árvore com todos os diffs usando `bk sccstool`. Se você ver alguns diffs estranhos ou código sobre o qual você tenha alguma dúvida, não hesite em enviar um e-mail para lista de email "internals" do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33. Além disso, se você acha que tem uma idéia melhor em como fazer algo, envie um email para o mesmo endereço com um patch. `bk diffs` irá produzir um patch para você após fazer as alterações no código fonte. Se você não tiver tempo para codificar sua idéia, apenas envie uma descrição.
10. **BitKeeper** tem um ótimo utilitário de ajudar que você pode acessar via `bk helptool`.
11. Note que qualquer commit (`bk ci` ou `bk citool`) irá disparar o envio da mensagem com as alterações para nossa lista de email internos, bem como a submissão openlogging.org usual apenas com os comentários da alteração. Geralmente você não precisar usar commit (já que o árvore pública não permitirá `bk push`), mas é preferível usar o método `bk diffs` descrito arteriormente.

Você também pode procurar alterações, comentários e código fonte online procurando por ex. `http://mysql.bkbits.net:8080/mysql-4.1` para MySQL 4.1.

O manual está em uma árvore separad que pode ser clonada com:

```
shell> bk clone bk://mysql.bkbits.net/mysqldoc mysqldoc
```

Existe também um árvore pública do BitKeeper para o MySQL Control Center e Connector/ODBC. Eles podem ser clonados da seguintes forma, respectivamente:

Para clonar o MySQL Control center, use o seguinte comando:

```
shell> bk clone http://mysql.bkbits.net/mysqlcc mysqlcc
```

Para clonar o Connector/ODBC, use o seguinte comando:

```
shell> bk clone http://mysql.bkbits.net/myodbc3 myodbc3
```

2.3.5 Lidando com Problemas de Compilação

Todos programas MySQL compilam de forma limpa sem alertas no solaris usando `gcc`. Em outros sistemas, alertas podem ocorrer devido a diferenças em arquivos include dos sistemas. Veja Seção 2.3.6 [MIT-pthreads], Página 106 para avisos que podem ocorrer usando MIT-pthreads. Para outros problemas, confira a lista abaixo.

A solução para vários problemas envolve reconfiguração. Se você precisa reconfigurar, faça notas do seguinte:

- Se `configure` é executado depois dele já ter sido chamado, ele pode usar informação que foi colhida durante a chamada anterior. Esta informação é armazenada no arquivo '`config.cache`'. Quando `configure` inicia, ele procura por este arquivo, lê seu conteúdo, se ele existir, assumindo que aquela informação continua correta. Essa conjetura é inválida quando você reconfigurar.
- Cada vez que você executa `configure`, você deve executar `make` de novo para recompilar. Entretanto, você pode desejar remover primeiro antigos arquivos objeto de construções anteriores, porque eles foram compilados usando diferentes opções de configuração.

Para prevenir antigas informações de configurações ou arquivos objetos de serem usados, execute estes comandos antes de re-executar `configure`:

```
shell> rm config.cache
shell> make clean
```

Uma alternativa, seria executar `make distclean`

A lista abaixo descreve alguns dos problemas compilando o MySQL que tem sido encontrados com mais frequencia:

- Se você obtêm erros quando '`sql_yacc.cc`' como os mostrados abaixo, você provavelmente tem de falta de memória ou espaço de swap:

  ```
  Internal compiler error: program cc1plus got fatal signal 11
    ou
  Out of virtual memory
    ou
  Virtual memory exhausted
  ```

  O problema é que `gcc` necessita de grande quantidade de memória para compilar '`sql_yacc.cc`' com funções inline. Tente executando `configure` com a opção `--with-low-memory`:

```
shell> ./configure --with-low-memory
```

Esta opção adiciona `-fno-inline` na a linha de compilação se você estiver usando `gcc` e `-O0` se você estiver usando outro programa. Você deve tentar a opção `--with-low-memory` mesmo se você tiver muita memória e espaço de swap que você ache ser suficieente para não ocorrer erros. Este problema tem ocorrido mesmo em sistemas com boas configurações de hardware e a opção `--with-low-memory` geralmente corrige isto.

- Por padrão, `configure` escolhe `c++` como o nome do compilador e GNU `c++` liga com `-lg++`. Se você estiver usando `gcc`, este comportamento pode causar problemas durante a compilação, como o seguinte:

```
configure: error: installation or configuration problem:
C++ compiler cannot create executables.
```

Você podem também ter problemas durante a compilação relacionados à `g++`, `libg++` ou `libstdc++`.

Uma causa destes problemas é que você pode não ter `g++` ou você pode ter `g++` mas não ter o `libg++` ou o `libstdc++`. De uma olhada no arquivo '`config.log`'. Ele deve conter a razão exata do porque seu compilador C++ não funciona! Para trabalhar evitando estes problemas, você pode usar `gcc` como seu compilador C++. Tente configurar a variável de ambiente `CXX` para `"gcc -O3"`. Por exemplo:

```
shell> CXX="gcc -O3" ./configure
```

Isto funciona porque `gcc` compila código fonte C++ tão bem quanto `g++` faz, mas não ifaz a ligação em `libg++` ou `libstdc++` por padrão.

Outra forma de corrigir estes problemas, com certeza, é instalando `g++`, `libg++` e `libstdc++`. No entanto gostariamos de lhe recomendar a não usar `libg++` ou `libstdc++` com o MySQL já que isto irá aumentar o tamanho do binário do mysqld sem lhe trazer nenhum benefício. Algumas versões destas bibliotecas também tem causado problemas estranhos para os usuários MySQL no passado.

Usar `gcc` como compilador C++ também é exigido, se você quiser compilar o MySQL com a funcionalidade RAID (veja Seção 6.5.3 [CREATE TABLE], Página 598 para mais informações sobre tipos de tabela RAID) e você estiver usando o GNU `gcc` versão 3 e acima. Se você obter erros como estes abaixo durante o estágio de ligação quando você configurar o MySQL para compilar com a opção `--with-raid`, tente usar o `gcc` como o seu compilador C++ definindo a variável de ambiente `CXX` mencionada acima:

```
gcc -O3 -DDBUG_OFF -rdynamic -o isamchk isamchk.o sort.o  libnisam.a
../mysys/libmysys.a ../dbug/libdbug.a ../strings/libmystrings.a -lpthread
-lz -lcrypt -lnsl -lm -lpthread
../mysys/libmysys.a(raid.o)(.text+0x79): In function `my_raid_create':
: undefined reference to `operator new(unsigned)'
../mysys/libmysys.a(raid.o)(.text+0xdd): In function `my_raid_create':
: undefined reference to `operator delete(void*)'
../mysys/libmysys.a(raid.o)(.text+0x129): In function `my_raid_open':
: undefined reference to `operator new(unsigned)'
../mysys/libmysys.a(raid.o)(.text+0x189): In function `my_raid_open':
: undefined reference to `operator delete(void*)'
../mysys/libmysys.a(raid.o)(.text+0x64b): In function `my_raid_close':
```

```
: undefined reference to 'operator delete(void*)'
collect2: ld returned 1 exit status
```

- Se sua compilação falhar com erros, como um dos seguintes, você deve atualizar sua versão de `make` para GNU `make`:

```
making all in mit-pthreads
make: Fatal error in reader: Makefile, line 18:
Badly formed macro assignment
  or
make: file 'Makefile' line 18: Must be a separator (:
  or
pthread.h: No such file or directory
```

  O Solaris e o FreeBSD são conhecidos por terem alguns problemas com o `make`.

  O GNU `make` versão 3.75 irá funcionar.

- Se você deseja definir algumas opções que devem ser usadas pelo seu compilador C ou C++, adicione as opções para as variáveis de ambiente `CFLAGS` e `CXXFLAGS`. Você pode também especificar os nomes do compilador a ser usado da mesma forma utilizando `CC` e `CXX`. Exemplo:

```
shell> CC=gcc
shell> CFLAGS=-O3
shell> CXX=gcc
shell> CXXFLAGS=-O3
shell> export CC CFLAGS CXX CXXFLAGS
```

  Olhe em Seção 2.2.8 [MySQL binaries], Página 86 para uma lista de definição de opções que tenham sido úteis em vários sistemas.

- Se você recebeu uma mensagem de erro como esta, é necessário atualizar o compilador `gcc`:

  O `gcc` 2.8.1 funciona, mas recomendamos o uso do `gcc` 2.95.2 ou `egcs` 1.0.3a em seu lugar.

- Se você obtem erros como estes vistos abaixo enquanto estiver compilando o `mysqld`, o `configure` não detectou corretamente o tipo do último argumento para `accept()`, `getsockname()` ou `getpeername()`:

```
cxx: Error: mysqld.cc, line 645: In this statement, the referenced
     type of the pointer value "&length" is "unsigned long", which
     is not compatible with "int".
new_sock = accept(sock, (struct sockaddr *)&cAddr, &length);
```

  Para corrigir isto, edite o arquivo '`config.h`' (que é gerado pelo `configure`). Procure por estas linhas:

```
/* Define as the base type of the last arg to accept */
#define SOCKET_SIZE_TYPE XXX
```

  Altere `XXX` para `size_t` ou `int`, dependendo de seu sistema operacional. (Perceba que você deverá fazer isto cada vez que você executar `configure`, porque `configure` regenera '`config.h`'.)

- O arquivo '`sql_yacc.cc`' é gerado pelo '`sql_yacc.yy`'. Normalmente o processo de construção não necessita criar '`sql_yacc.cc`', porque o MySQL já vem com uma cópia pré-gerada. Entretanto, se você necessita recriá-lo você pode encontrar este erro:

  ```
  "sql_yacc.yy", line xxx fatal: default action causes potential...
  ```

  Isto é um indício de que sua versão do `yacc` é deficiente. Provavelmente você precisará instalar o `bison` (a versão GNU de `yacc`) e usá-lo no lugar do `yacc`.
- Se você necessita depurar `mysqld` ou um cliente MySQL, execute `configure` com a opção `--with-debug`, então recompile e ligue seus clientes com a nova biblioteca cliente. Veja Seção E.2 [Debugging client], Página 1085.
- Se você tem um erro de compilação no Linux (ex. SuSE Linux 8.1 ou Red Hat Linux 7.3) parecido com o seguinte:

  ```
  libmysql.c:1329: warning: passing arg 5 of 'gethostbyname_r' from incompatible p
  libmysql.c:1329: too few arguments to function 'gethostbyname_r'
  libmysql.c:1329: warning: assignment makes pointer from integer without a cast
  make[2]: *** [libmysql.lo] Error 1
  ```

  Por padrão, o script `configure` tenta determinar o número correto de argumentos usando o compilador GNU C++ `g++`. Ele testa os resultados errados permitidos, se o `g++` não está instalado. Existem dois modos de contornar este problema:
  - Certifique-se de que o GNU C++ `g++` está instalado. Em algumas distribuições Linux, o pacote exigido é chamado `gpp`, em outro ele é chamado `gcc-c++`.
  - Use o `gcc` como o seu compilador C++ configurando a variáavel de ambiente `CXX` para `gcc`:

    ```
    export CXX="gcc"
    ```

  Note que você precisa executar o `configure` novamente após isto.

2.3.6 Notas MIT-pthreads

Esta seção descreve alguns dos detalhes envolvidos no uso de MIT-pthreads.

Note que no Linux você `NÃO` deve usar MIT-pthreads mas instalar LinuxThreads! Veja Seção 2.6.2 [Linux], Página 137.

Se seu sistema não fornece suporte nativo a thread, você precisará construir o MySQL usando o pacote MIT-pthreads. Isto inclui antigos sistemas FreeBSD, SunOS 4.X, Solaris 2.4 e anteriores entre outros. Veja Seção 2.2.3 [Qual SO], Página 78.

Note que a partir do MySQL 4.0.2, MIT-pthreads não fazem mais parte da distribuição fonte. Se você precisar deste pacote, você precisa fazer o download dele separadamente em `http://www.mysql.com/Downloads/Contrib/pthreads-1_60_beta6-mysql.tar.gz`

Depois do download, extraia este arquivo fonte no nível mais alto do diretório de fontes do MySQL. Ele criará um novo subdiretório `mit-pthreads`.

- Na maioria dos sitemas, você pode forçar o uso de MIT-pthreads executando o `configure` com a opção `--with-mit-threads`:

  ```
  shell> ./configure --with-mit-threads
  ```

  Construção em um diretório não fonte não é suportado com o uso de MIT-pthreads, porque nós queremos minimizar nossas alterações para este código.

- As verificações que determinam se MIT-pthreads será usado ou não, ocorrerá somente durante a parte do processo de configuração que trata com o código do servidor. Se você configurou a distribuição usando `--without-server` para construir somente o código cliente, clientes não irão saber se o MIT-pthreads está sendo usado e irá usar conexões socket Unix por padrão. Como os sockets Unix não funcionam sob MIT-pthreads, isto significa que você precisará usar `-h` ou `--host` quando executar programas clientes.
- Quando o MySQL é compilado usando MIT-pthreads, travas de sistema são desabilitadas por padrão por razões de performance. Você pode dizer ao servidor para usar travas de sistema com a opção `--external-locking`. Isto só é necessário se você quiser executar dois servidores MySQL no mesmo diretório de dados (no que não é recomendado)
- Algumas vezes o comando pthread `bind()` falha ao ligar a um socket sem nenhuma mensagem de erro (pelo menos no Solaris). O resultado é que todas conexões ao servidor falham. Por exemplo:

  ```
  shell> mysqladmin version
  mysqladmin: connect to server at '' failed;
  error: 'Can't connect to mysql server on localhost (146)'
  ```

  A solução para isto é matar o servidor `mysqld` e reiniciá-lo. Isto só aconteceu conosco quando forçamos uma queda do servidor e fizemos uma reinicialização imediata.
- Com MIT-pthreads, a chamada de sistema `sleep()` não é interrompível com `SIGINT` (break). Isto só é percebido quando você executa `mysqladmin --sleep`. Você deve esperar pela chamada `sleep()` para terminar, antes da interrução ser servida e o processo parar.
- Na ligação, você pode receber mensagens de alerta como estes (pelo menos no Solaris); elas podem ser ignoradas:

  ```
  ld: warning: symbol `_iob' has differing sizes:
      (file /my/local/pthreads/lib/libpthread.a(findfp.o) value=0x4;
  file /usr/lib/libc.so value=0x140);
      /my/local/pthreads/lib/libpthread.a(findfp.o) definition taken
  ld: warning: symbol `__iob' has differing sizes:
      (file /my/local/pthreads/lib/libpthread.a(findfp.o) value=0x4;
  file /usr/lib/libc.so value=0x140);
      /my/local/pthreads/lib/libpthread.a(findfp.o) definition taken
  ```

- Alguns outros alertas também podem ser ignorados:

  ```
  implicit declaration of function `int strtoll(...)'
  implicit declaration of function `int strtoul(...)'
  ```

- Não colocamos `readline` para funcionar com MIT-pthreads. (Isto não é necessário, mas pode ser interessante para alguns.)

2.3.7 Instalando o MySQL a partir do Fonte no Windows

Estas instruções descrevem como construir o binário do MySQL a partir do fonte paras versões 4.1 e acima no Windows. As instruções são fornecidas para construir binários a partir de uma distribuição fonte padrão ou a partir da árvore do BitKeeper que contém o fonte do desenvolvimento mais atuais.

Nota: As instruções neste documento estão restritas aos usuários que queiram testar o MySQL no Windows a partir da última distribuição fonte ou da árvore do BitKeeper. Para uso em produção, a MySQL AB não aconselha que você utilize um servidor MySQL construído por você mesmo a partir de um fonte. Normalmente é melhor usar uma distribuição binária precompilada do MySQL que é construída especificamente para desempenho otimizado no Windows pela MySQL AB. Instruções para instalar uma distribuição binária está disponível em Seção 2.1.1 [Windows installation], Página 60.

Para construir o MySQL no Windows a partir do fonte, você precisa dos seguintes compiladores e recursos disoníveis em seu sistema Windows:

- Compilador VC++ 6.0 (atualizado com o SP 4 ou 5 e pacote Pre-processador) O pacote Pre-processador é necessário para a macro assembler. Mais detalhes em: `http://msdn.microsoft.com/vstudio/downloads/updates/sp/vs6/sp5/faq.aspx`.
- Aproximadamente 45 MB de espaço em disco.
- 64 MB de RAM

Você também precisará de um distribuição fonte para o Windows. Existem dois modos de conseguir uma distribuição fonte do MySQL versão 4.1 e acima:

1. Obtenha um pacote de uma distribuição fonte pela MySQL AB para a versão do MySQL que você está particularmente interessado. Distribuições fontes empacotadas estão disponíveis para versões distribuídas do MySQ e podem ser obtidas em `http://www.mysql.com/downloads/`.
2. Você pode empacotar um distribuição fonte você mesmo a partir da ultima árvore fonte de desenvolvimento do BitKeeper. Se você planeja fazer isto, você deve criar o pacote em um sistema Unix e então transfrí-lo para seu sistema Windows. (A razão para isto é que alguns dos passos de configuração e construção exigem ferramentas que funcionam apenas no Unix.) A abordagem do BitKeeper, exige:
   - Um sistema executando Unix ou um sistema tipo Unix, como o Linux
   - BitKeeper 3.0 instalado neste sistema. Você pode obter o BitKeeper em `http://www.bitkeeper.com/`.

Se você estiver usando uma distribuição fonte do Windows, você pode ir diretamente para Seção 2.3.7.1 [Windows VC++ Build], Página 108. Para contruir a partir da árvore do BitKeeper, vá para Seção 2.3.7.2 [Windows BitKeeper Build], Página 110.

Se você encontrar alguma coisa que não está funcionando como esperado, ou tiver sugestões sobre o mode de melhorar o processo de construção atual no Windows, envie uma mensagem para a lista de email `win32`. Veja Seção 1.7.1.1 [Mailing-list], Página 33.

2.3.7.1 Construindo o MySQL Usando VC++

Nota: O MySQL 4.1 e arquivos do espeço de trabalho do VC++ são compatíveis com o Microsoft Visual Studio 6.0 e as edições acima (7.0/.NET) e testados pela equipe da MySQL AB antes de cada distribuição.

Siga este procedimento para construir o MySQL:

1. Crie um diretório de trabalho (ex.: '`workdir`').
2. Descompacte a distribuição fonte no diretório mencionado acima usando `Winzip` ou outra ferramenta que possa ler arquivos '`.zip`'.

3. Inicie o compilador VC++ 6.0.
4. No menu `File`, selecione `Open Workspace`.
5. Abra o workspace '`mysql.dsw`' que você encontrar no diretório de trabalho.
6. No menu `Build`, selcione o menu `Set Active Configuration`.
7. Clique sobre a tela selecionada `mysqld - Win32 Debug` e clique OK.
8. Pressione `F7` para iniciar a construção da depuração do servidor, bibliotecas e alguns aplicativos clientes.
9. Compile as versões distribuídas que você desejar, do mesmo modo.
10. Versões depuradas dos programas e bibliotecas são colocados nos diretórios '`client_debug`' e '`lib_debug`'. Versões liberadas dos programas e bibliotecas são colocados nos diretórios '`client_release`' e '`lib_release`'. Note que se você quiser construir tanto versões liberadas quanto depuradas você pode selecionar a opção "build all" do menu `Build`.
11. Teste o servidor. O servidor construído usando as instruções anteriores irá esperar que o diretório base e de dados do MySQL seja '`C:\mysql`' e '`C:\mysql\data`' por padrão. Se você quiser testar o seu servidor usando o diretório raiz de uma árvore fonte e seu diretório de dados como o diretório base e o diretório de dados, você precisará dizer ao servidor os seus caminhos. Você também pode fazer into na linha de comando com as opções `--basedir` e `--datadir`, ou colocar opções apropriadas no arquivo de opções (o arquivo '`C:\my.cnf`' ou '`my.ini`' no diretório do Windows). Se você tiver um diretório de dados existente em qualquer lugar que você queira usar, você pode especificá-lo no se caminho.
12. Inicie o ser servidor a partir do diretório '`client_release`' ou '`client_debug`', dependendo de qual servidor você queira usar. O instruções gerais de inicializaão do servidor estão em Seção 2.1.1 [Windows installation], Página 60. Você precisará adaptar as instruções de forma apropriada se você quiser usar um diretório base ou diretório de dados diferente.
13. Quando o servidor está em execução de modo independente ou como um serviço daseado em sua configuração, tente se conectar a ele pelo utilitário interativo `mysql` de linha de comando que existe em seu diretório '`client_release`' ou '`client_debug`'.

Quando você estiver certo de que os programas que você construiu estão funcionando corretamente, pare o servidor. Então instale o MySQL da seguinte forma:

1. Crie o diretório para instalar os arquivos do MySQL. Por exemplo, para instalar dentro de '`C:\mysql`'), use estes comandos:

```
C:
mkdir \mysql
mkdir \mysql\bin
mkdir \mysql\data
mkdir \mysql\share
mkdir \mysql\scripts
```

   Se você quiser compilar outros clientes e ligá-los ao MySQL, você também deve criar diversos diretórios adicionais:

```
mkdir \mysql\include
```

```
mkdir \mysql\lib
mkdir \mysql\lib\debug
mkdir \mysql\lib\opt
```

Se você quiser fazer um benchamrk do MySQL, crie este diretório:

```
mkdir \mysql\sql-bench
```

Benchmark exigem suporte Perl.

2. Copie do diretório `workdir` para o diretório `c:\mysql` os seguintes diretórios:

```
copy client_release\*.exe C:\mysql\bin
copy client_debug\mysqld.exe C:\mysql\bin\mysqld-debug.exe
xcopy scripts\*.* C:\mysql\scripts /E
xcopy share\*.* C:\mysql\share /E
```

Se você quiser compilar outros clientes e ligá-los ao MySQL, você também deve fazer isto:

```
copy lib_debug\mysqlclient.lib C:\mysql\lib\debug
copy lib_debug\libmysql.* C:\mysql\lib\debug
copy lib_debug\zlib.* C:\mysql\lib\debug
copy lib_release\mysqlclient.lib C:\mysql\lib\opt
copy lib_release\libmysql.* C:\mysql\lib\opt
copy lib_release\zlib.* C:\mysql\lib\opt
copy include\*.h C:\mysql\include
copy libmysql\libmysql.def C:\mysql\include
```

Se você quiser fazer um benchmark do MySQL, você também deve fazer isto:

```
xcopy sql-bench\*.* C:\mysql\bench /E
```

Configure e inicie o servidor da mesma forma que a distribuição binária do Windows. Veja Seção 2.1.1.3 [Windows prepare environment], Página 62.

2.3.7.2 Criando um Pacote Fonte do Windows a partir da Última Fonte de Desenvolvimento

Para construir o último pacote fonte do Windows a partir da arvoré fonte atual do BitKeeper, use as seguintes instruções. Por favor, note que este procedimento deve ser realizado em um sistema executando um sistema opercional Unix ou similar. (Sabe-se que este procedimento funciona bem com o Linux, por exemplo.)

1. Clone a árvore fonte do BitKeeper para o MySQL (versão 4.1 ou acima, como desejado). Para mais informações sobre como clonar a árvore fonte veja as instruções em Seção 2.3.4 [Installing source tree], Página 100.
2. Configure e construa as distribuições para que você tenha um binário do servidor para trabalhar. Um modo de se fazer isto é executar o seguinte comando no diretório de mais alto nível de sua árvore fonte:

```
shell> ./BUILD/compile-pentium-max
```

3. Depois de se certificar que o processo de construção foi completado com sucesso, execute o seguinte script utilitário a a partir do diretório de nível mais alto da sua arvore fonte:

```
shell> ./scripts/make_win_src_distribution
```

Este script cria um pacote fonte Windows. para ser usado em seu sistema Windows. Você pode fornecer diferentes opções para o script baseado em suas necessidades. Ele aceita as seguintes opções:

```
--debug   Depura, sem criar o pacote
--tmp     Especifica a localização temporária
--suffix  Nome de sufixo para o pacote
--dirname Nome do diretório onde os arquivos são copiados (intermediario)
--silent  Não apresenta uma lista dos arquivos processados
--tar     Cria um pacote tar.gz em vez de .zip
--help    Mostra esta mensagem de ajuda
```

Por padrão, `make_win_src_distribution` cria um arquivo zipado com o nome '`mysql-VERSION-win-src.zip`', onde `VERSION` representa a versão de sua árvore fonte do MySQL.

4. Faça uma copia ou upload para a sua máquina o pacote fonte Windows que você tiver criado. Para compilá-lo use as instruções em Seção 2.3.7.1 [Windows VC++ Build], Página 108.

2.4 Configurações e Testes Pós-instalação

Uma vez instalado o MySQL (de uma distribuição binária ou fonte), você deve inicializar as tabelas de concessões, iniciar o servidor e ter certeza que o servidor está funcionando bem. Você pode também desejar que o servidor inicie e pare automaticamente quando seu sistema iniciar e desligar.

Normalmente você instala as tabelas de concessões e inicia o servidor assim para instalações baseadas em uma distribuição fonte:

```
shell> ./scripts/mysql_install_db
shell> cd diretorio_instalação_mysql
shell> ./bin/mysqld_safe --user=mysql &
```

Para uma distribuição binária (sem ser pacotes RPM ou PKG), faça isto:

```
shell> cd diretorio_instalação_mysql
shell> ./bin/mysql_install_db
shell> ./bin/mysqld_safe --user=mysql &
```

O script `mysql_install_db` cria o banco de dados `mysql` que irá armazenar todos privilégios do banco de dados, o banco de dados `test` que você poderá usar para testar o MySQL e também entradas de privilégio para o usuário que usa o `mysql_install_db` e o usuário `root`. As estrandas são criadas sem senhas. O script `mysqld_safe` inicia o servidor `mysqld`. (Se sua versão for anterior a 4.0, use `safe_mysqld` em vez de `mysqld_safe`.)

`mysql_install_db` não irá sobrescrever nenhuma tabela de privilégios antiga, então deve ser seguro executá-lo em quaisquer circunstâncias. Se você não deseja ter o banco de dados `test` você pode removê-lo com `mysqladmin -u root drop test` depois de iniciar o servidor.

Testes são geralmente facilmente feitos de um diretório raiz da distribuição MySQL. Para uma distribuição binária, este é seu diretório de instalação (normalmente algo como

'`/usr/local/mysql`'). Para uma distrubuição fonte, este é o diretório principal da sua árvore fonte do MySQL.

Nos comandos mostrados abaixo nesta seção e nas seguintes subseções, `BINDIR` é o caminho para a localização na qual os programas como `mysqladmin` e `mysqld_safe` estão instalados. Para uma distribuição binária este é o diretório '`bin`'. Para uma distribuição fonte, `BINDIR` é provavelmente '`/usr/local/bin`', a menos que você especifique um diretório de instalação diferente de '`/usr/local`' quando você executa `configure`. `EXECDIR` é a localização na qual o servidor `mysqld` está instalado. Para uma distribuição binária, isto é o mesmo que `BINDIR`. Para uma distribuição fonte, `EXECDIR` é provavelmente '`/usr/local/libexec`'.

Os testes são descritos em detalhes abaixo:

1. Se necessário, inicie o servidor `mysqld` e configure as tabelas de concessões iniciais contendo os privilégios que determinam como os usuários estão permitidos a conectar ao servidor. Isto é feito normalmente com o script `mysql_install_db`:

   ```
   shell> scripts/mysql_install_db
   ```

   Normalmente, `mysql_install_db` precisa ser executado somente na primeira vez que você instala o MySQL. Portanto, se você estiver atualizando uma instalação existente, você pode pular este passo. (entretanto, `mysql_install_db` é realmente seguro de usar e não irá atualizar nenhuma tabela que já exista, então se você não tem certeza do que fazer, você pode sempre executar `mysql_install_db`.)

   `mysql_install_db` cria seis tabelas (`user`, `db`, `host`, `tables_priv`, `columns_priv` e `func`) no banco de dados `mysql`. Uma descrição dos privilégios iniciais é fornecido em Seção 4.4.4 [Default privileges], Página 260. De forma resumidao, estes privilégios permitem que o usuário root faça qualquer coisa no MySQL, e permitem a qualquer um a criar ou usar bancos de dados com o nome de `'test'` ou iniciando com `'test_'` .

   Se você não configurar as tabelas de concessões, o seguinte erro irá aparecer no arquivo log quando você não iniciar o servidor:

   ```
   mysqld: Can't find file: 'host.frm'
   ```

   O erro acima pode também ocorrer com uma distribuição binária do MySQL se você não iniciar o MySQL executando o `./bin/mysqld_safe`! Veja Seção 4.8.2 [`mysqld_safe`], Página 331.

   Você deve precisar executar `mysql_install_db` como `root`. Entretanto, se você preferir, pode executar o servidor MySQL como um usuário (não-`root`) sem privilégios, desde que o usuário possa ler e escrever arquivos no diretório de banco de dados. Instruções para executar o MySQL como um usuário sem privilégios é detalhado em Seção A.3.2 [Alterando usuários MySQL], Página 923

   Se você tiver problemas com o `mysql_install_db`, veja Seção 2.4.1 [`mysql_install_db`], Página 115.

   Existem algumas alternativas para executar o script `mysql_install_db` como ele é fornecido na distribuição MySQL:

   - Você pode querer editar o `mysql_install_db` antes de executá-lo, para alterar os privilégios iniciais que são instalados nas tabelas de concessões. Isto é útil se você deseja instalar o MySQL em várias máquinas com os mesmos privilégios. Neste caso, é provável que você só precise adicionar algumas instruções `INSERT` extras para as tabelas `mysql.user` e `mysql.db`.

- Se você deseja alterar o conteúdo da tabelas de concessões depois de instalá-las, você pode executar `mysql_install_db`, então usar `mysql -u root mysql` para conectar às tabelas de concessões como o usuário `root` e usar instruções SQL para modificá-las diretamente.
- É possível recriar as tabelas de permissões completamente depois delas já terem sido criadas. Você pode querer fazer isto se você já instalou as tabelas mas deseja recriá-las depois das edições `mysql_install_db`.

Para maiores informações sobre estas alternativas, veja Seção 4.4.4 [Default privileges], Página 260.

2. Inicie o servidor MySQL assim:

```
shell> cd diretorio_instalacao_mysql
shell> bin/mysqld_safe &
```

Se a sua versão do MySQL for mais antiga do que 4.0, substitua `bin/safe_mysqld` por `bin/mysqld_safe` no comando:

Se você tiver problemas iniciando o servidor, veja Seção 2.4.2 [Starting server], Página 116.

3. Use `mysqladmin` para verificar se o servidor está em execução. Os seguintes comandos fornecem um teste simples para conferir se o servidor está em funcionamento e respondendo às conexões:

```
shell> BINDIR/mysqladmin version
shell> BINDIR/mysqladmin variables
```

A saída de `mysqladmin version` varia muito pouco dependendo de sua plataforma e versão do MySQL, mas deve ser similar a esta mostrada abaixo:

```
shell> BINDIR/mysqladmin version
mysqladmin  Ver 8.14 Distrib 3.23.32, for linux on i586
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license.

Server version          3.23.32-debug
Protocol version        10
Connection              Localhost via Unix socket
TCP port                3306
UNIX socket             /tmp/mysql.sock
Uptime:                 16 sec

Threads: 1  Questions: 9  Slow queries: 0
Opens: 7  Flush tables: 2  Open tables: 0
Queries per second avg: 0.000
Memory in use: 132K  Max memory used: 16773K
```

Para ter uma idéia do que você pode fazer com `BINDIR/mysqladmin`, invoque-o com a opção `--help`.

4. Verifique se você pode desligar o servidor:

```
shell> BINDIR/mysqladmin -u root shutdown
```

5. Verifique que você possa reiniciar o servidor. Faça isto usando `mysqld_safe` ou chamado o `mysqld` diretamente. Por exemplo:

```
shell> BINDIR/mysqld_safe --log &
```

Se o `mysqld_safe` falhar, tente executá-lo do diretório de instalação do MySQL (se você já não estiver lá). Se não funcionar, veja Seção 2.4.2 [Starting server], Página 116.

6. Execute alguns testes básicos para verificar se o servidor está funcionando. A saída deve ser similar ao mostrado abaixo:

```
shell> BINDIR/mysqlshow
+-----------+
| Databases |
+-----------+
| mysql     |
+-----------+

shell> BINDIR/mysqlshow mysql
Database: mysql
+--------------+
|    Tables    |
+--------------+
| columns_priv |
| db           |
| func         |
| host         |
| tables_priv  |
| user         |
+--------------+

shell> BINDIR/mysql -e "SELECT host,db,user FROM db" mysql
+------+--------+------+
| host | db     | user |
+------+--------+------+
| %    | test   |      |
| %    | test_% |      |
+------+--------+------+
```

Também existe uma suite de benchmark no diretório '`sql-bench`' (sob o diretório de instalação do MySQL) que você pode usar para comparar como o MySQL se comporta em diferentes plataformas. O diretório '`sql-bench/Results`' contém os resultados de várias execuções em diferentes bancos de dados e plataformas. Os seguintes módulos Perl adicionais são necessários para executar o pacote de benchamrk:

```
DBI
DBD-mysql
Data-Dumper
Data-ShowTable
```

Estes módulos podem ser obtidos em CPAN `http://www.cpan.org/`. Veja Seção 2.7.1 [Instalação Perl], Página 164.

O diretório '`sql-bench/Results`' contém os resultados de várias execuções em diferentes bancos de dados e plataformas. Para executar todos testes, execute estes comandos:

```
shell> cd sql-bench
shell> run-all-tests
```

Se você não possui o diretório '`sql-bench`', você provavelmente está usando uma distribuição binária RPM. (Distribuições fontes RPMs incluem o diretório com os benchmarks.) Neste caso, você deve primeiramente instalar a suite de benchmark antes de poder usá-lo. A partir da versão 3.22 do MySQL, começaram a existir arquivos RPMs de benchmark chamados '`mysql-bench-VERSION-i386.rpm`' que contém código ie dados de benchmark.

Se você tem uma distribuição fonte, você também pode executar os testes no subdiretório '`tests`'. Por exemplo, para executar '`auto_increment.tst`', faça isto:

```
shell> BINDIR/mysql -vvf test < ./tests/auto_increment.tst
```

Os resultados esperados são mostrados no arquivo '`./tests/auto_imcrement.res`'.

2.4.1 Problemas Executando o `mysql_install_db`

O propósito do script `mysql_install_db` é gerar novas tabelas de privilégios. Ele não irá afeter nenhum outro dado! Ele também não fará nada se você já tem a tabela de privilégio do MySQL instalada.

Se você deseja refazer suas tabelas de privilégios, você deve desligar o servidor `mysqld`, se ele já está executando, então faça assim:

```
mv diretorio-dados-mysql/mysql diretorio-dados-mysql/mysql-old
mysql_install_db
```

Esta seção relaciona alguns problemas que podem ser encontrados ao executar `mysql_install_db`:

`mysql_install_db` não instala as tabelas de permissões

Você pode descobrir que o `mysql_install_db` falha ao instalar as tabelas de permissões e termina depois de mostrar as seguintes mensagens:

```
starting mysqld daemon with databases from XXXXXX
mysql daemon ended
```

Neste caso, você deve examinar o arquivo de log com muito cuidado! O log deve se encontrar no diretório '`XXXXXX`' nomeado pela mensagem de erro, e deve indicar porque `mysqld` não inicializa. Se você não entende o que aconteceu, inclua o log quando você postar um relato de erro usando `mysqlbug`! Veja Seção 1.7.1.3 [Bug reports], Página 36.

Já existe um daemon `mysqld` sendo executado

Neste caso, provavelmente não será necessário executar o `mysql_install_db`. Você deve executar o `mysql_install_db` somente uma vez, quando você instalar o MySQL da primeira vez.

Instalair um segundo daemon `mysqld` não funciona quando um daemon

estiver em execução.

Isto pode acontecer quando você já tiver uma instalação do MySQL existente, mas deseja colocar uma nova instalação em um diferente lugar (por exemplo, para testes, ou talvez você simplesmente deseja executar duas instalações ao mesmo tempo). Geralmente o problema que ocorre quando você tenta executar o segundo servidor é que ele tenta usar o mesmo socket e porta que o outro. Neste caso você irá obter a mensagem de erro: `Can't start server: Bind on TCP/IP port: Address already in use` ou `Can't start server: Bind on unix socket...`. Veja Seção 4.2 [Múltiplos servidores], Página 219.

Você não tem direito de escrita no diretório '`/tmp`'

Se você não tem direito de escrita para criar um arquivo socket no local padrão (em '`/tmp`') ou permissão para criar arquivos temporáris em '`/tmp,`' você irá obter um erro quando executar `mysql_install_db` ou quando iniciar ou usar `mysqld`.

Você pode especificar socket e diretório temporário diferentes, como segue:

```
shell> TMPDIR=/algum_dir_tmp/
shell> MYSQL_UNIX_PORT=/algum_dir_tmp/mysqld.sock
shell> export TMPDIR MYSQL_UNIX_PORT
```

Veja Seção A.4.5 [Problems with mysql.sock], Página 929.

'`algum_dir_tmp`' deve ser o caminho para o mesmo diretório no qual você tem permissão de escrita. Veja Apêndice F [Environment variables], Página 1092.

Depois disto você deve estar apto para executar `mysql_install_db` e iniciar o servidor com estes comandos:

```
shell> scripts/mysql_install_db
shell> BINDIR/mysqld_safe &
```

`mysqld` falha imediatamente

Se você estiver executando RedHat Versão 5.0 com uma versão de `glibc` anterior a 2.0.7-5 você deve ter certeza que você instalou todos os patches para a `glibc`! Existe muita informação sobre isto nos arquivos das listas de mensagens do MySQL. Links para os arquivos de correio estão disponíveis online em `http://lists.mysql.com/`. Veja também Seção 2.6.2 [Linux], Página 137.

Você pode também iniciar o `mysqld` manualmente usando a opção `--skip-grant-tables` e adicionar a informação de privilégios usando o `mysql`:

```
shell> BINDIR/mysqld_safe --skip-grant-tables &
shell> BINDIR/mysql -u root mysql
```

Do `mysql`, execute manualmente os comandos SQL em `mysql_install_db`. Tenha certeza de executar `mysqladmin flush_privileges` ou `mysqladmin reload` após dizer ao servidor para recarregar as tabelas de permissões.

2.4.2 Problemas Inicializando o Servidor MySQL

Se você for usar tabelas que suportem transações (BDB, InnoDB), primeiro deve-se criar um arquivo my.cnf e configurar opções de inicialização para os tipos de tabelas que você planeja usar. Veja Cap"ptexi tulo 7 [Table types], Página 630.

Geralmente, você inicia o servidor `mysqld` de uma das três maneiras:

- Invocando `mysql.server`. Este script é usado primariamente na inicialização e finalização do sistema, e é descrito de forma mais completa em Seção 2.4.3 [Automatic start], Página 118.
- Invocando `mysqld_safe`, que tenta determinar as opções apropriadas para `mysqld` e então executá-lo com estas opções. Veja Seção 4.8.2 [`mysqld_safe`], Página 331.
- Para o Windows NT/2000/XP, veja Seção 2.1.1.7 [NT start], Página 66.
- Invocando o `mysqld` diretamente.

Quando o daemon `mysqld` inicia, ele altera o diretório para o diretório de dados. É neste diretório que ele espera gravar arquivos de log e o arquivo pid (com o ID do processo) e onde ele espera encontrar os bancos de dados.

A localização do diretório de dados é especificada quando a distribuição é compilada. Entretanto, se o `mysqld` espera encontrar o diretório de dados em lugar diferente de onde ele realmente está no seu sistema, ele não funcionará corretamente. Se você tiver problemas com caminhos incorretos você pode encontrar quais opções o `mysqld` permite e quais são as configurações do caminho padrão chamando o `mysqld` com a opção `--help`. Você pode sobrescrever os padrões especificando os caminhos corretos como argumentos de linha de comando ao `mysqld`. (Estas opções também podem ser usadas com o `mysqld_safe`).

Normalmente você precisaria indicar ao `mysqld` somente o diretório base sob o qual o MySQL é instalado. Você pode fazer isso usando a opção `--basedir`. Você pode também usar `--help` para conferir o efeito das opeções para se alterar o caminho (perceba que `--help` *deve* ser a opção final do comando `mysqld`. Por exemplo:

```
shell> EXECDIR/mysqld --basedir=/usr/local --help
```

Uma vez que você determina as configurações de caminho que você deseja, inicie o servidor sem a opção `--help`.

Qualquer que tenha sido o método utilizado para iniciar o servidor, se houver falha na inicialização, confira o arquivo de log para ver se você pode entender o porquê. Arquivos log estão localizados no diretório dados (normalmente '`/usr/local/mysql/data`' para uma distribuição binária, '`/usr/local/var`' para uma distribuição fonte, '`\mysql\data\mysql.err`' no Windows.) Procure no diretório de dados por arquivos com nomes no formato '`nome_maquina.err`' e '`nome_maquina.log`' onde `nome_maquina` é o nome do servidor. Então confira as últimas linhas destes arquivos:

```
shell> tail nome_maquina.err
shell> tail nome_maquina.log
```

Se você encontrar algo como o seguinte no arquivo log:

```
000729 14:50:10  bdb:  Recovery function for LSN 1 27595 failed
000729 14:50:10  bdb:  warning: ./test/t1.db: No such file or directory
000729 14:50:10  Can't init databases
```

Significa que você não inicializou o `mysqld` com `--bdb-no-recover` e o Berkeley DB encontrou algo errado com seus arquivos log quando ele tentou recuperar seus bancos de dados. Para poder continuar, você deve mover o antigo arquivo log Berkeley DB do diretório do banco de dados para outro lugar, onde poderá examiná-los posteriormente. Os arquivos log são nomeados '`log.0000000001`', onde o número irá incrementar com o tempo.

Se você estiver executando o `mysqld` com suporte a tabelas BDB e o `mysqld` falhar no início, pode ser devido a alguns problemas com o arquivo de recuperação BDB. Neste caso

você pode tentar iniciar o `mysqld` com `--bdb-no-recover`. Se isto ajudar, então você pode remover todos os arquivos '`log.*`' do diretório de dados e tentar iniciar o `mysqld` novamente.

Se você obter o seguinte erro, significa que algum outro programa (ou outro servidor `mysqld`) já está usando a porta TCP/IP ou socket `mysqld` está tentando usar:

```
Can't start server: Bind on TCP/IP port: Address already in use
  ou
Can't start server: Bind on unix socket...
```

Use `ps` para ter certeza que você não tem outro servidor `mysqld` em execução. Se você não consegue encontrar outro servidor, você pode tentar executar o comando `telnet sua_maquina numero_porta_tcp-ip` e apertar `ENTER` várias vezes. Se você não obter uma mensagem como `telnet: Unable to connect to remote host: Connection refused`, algo está usando a mesma porta TCP/IP que o `mysqld` está tentando usar. Veja Seção 2.4.1 [mysql_install_db], Página 115 e Seção 4.2 [Multiple servers], Página 219.

Se o `mysqld` está atualmente em execução, você pode verificar as configurações que ele está usando executando este comando:

```
shell> mysqladmin variables
```

ou

```
shell> mysqladmin -h 'your-host-name' variables
```

Se você obter o `Errcode 13`, que significa `Permission denied`, ao iniciar o `mysqld` isto significa que você não pode ter o direito de leitura/criação de arquivos no diretório do banco de dados ou log. Neste caso você também deve iniciar o `mysqld` como usuário `root` ou alterar a permissão para os arquivos e diretórios envolvidos para uqe você tenha o direito de usá-los.

Se o `mysqld_safe` inicia o servidor mas você não consegue se conectar a ele, tenha certeza que você tem uma entrada no arquivo '`/etc/hosts`' que parece com isto:

```
127.0.0.1       localhost
```

Este problema só ocorre em sistemas que não possuem uma biblioteca thread funcional e para o qual o MySQL deve estar configurado para usar MIT-pthreads.

Se você não consegue iniciar o `mysqld` você pode tentar criar um arquivo para rastreamento de erros (trace) para encontrar o problema. Veja Seção E.1.2 [Making trace files], Página 1080.

Se você estiver utilizando tabelas InnoDB, procure pelas opções especificas de inicialização do InnoDB. Veja Seção 7.5.3 [InnoDB start], Página 644.

Se você estiver usando tabelas BDB (Berkeley DB), você deve se familiarizar com as diferentes opções especificas de inicialização do BDB. Seção 7.6.3 [BDB start], Página 699.

2.4.3 Inicializando e parando o MySQL automaticamente.

Os scripts `mysql.server` e `mysqld_safe` podem ser usados para iniciar o servidor automaticamente na inicialização do sistema. `mysql.server` também pode ser usado para parar o servidor.

O script `mysql.server` pode ser usado para inicializar ou parar o servidor utilizando-o com os argumentos `start` ou `stop`:

```
shell> mysql.server start
shell> mysql.server stop
```

`mysql.server` pode ser encontrado no diretório '`share/mysql`' sob o diretório de instalação do MySQL ou no diretório '`support-files`' da árvore fonte do MySQL.

Note que se você usa o pacote RPM do Linux (`MySQL-server-VERSÃO.rpm`), o script `mysql.server` já estará instalada como '`/etc/init.d/mysql`' - você não precisa instalá-lo manualmente. Veja Seção 2.1.2 [Linux-RPM], Página 69 para mais informações sobre pacotes RPM Linux.

No Mac OS X, você pode instalar um pacote do MySQL Startup Item separado para habilitar a inicialização automática do MySQL no boot so sistema. Veja Seção 2.1.3 [Mac OS X installation], Página 71 para maiores detalhes.

Antes do `mysql.server` iniciar o servidor, ele vai para o diretório de instalação do MySQL, e então chama o `mysqld_safe`. Você pode precisar editar o `mysql.server` se tiver uma distribuição binária instalada em um local não-padrão. Modifique-o para chamar o diretório (`cd`) apropriado antes de executar o `safe_mysql`. Se você deseja que o servidor seja executado com um usuário específico, adicione uma linha `user` apropriada para o arquivo '`/etc/my.cnf`', como será visto posteriormente nesta seção.

`mysql.server stop` desliga o servidor MySQL enviando um sinal para ele. Você pode desligar o servidor manualmente executando `mysqladmin shutdown`.

Você precisa adicionar estes comandos start e stop nos lugares apropriados de seus arquivos '`/etc/rc.*`' quando você quiser iniciar o MySQL automaticamente no seu servidor.

On most current Linux distributions, it is sufficient to copy the file `mysql.server` into the '`/etc/init.d`' directory (or '`/etc/rc.d/init.d`' on older Red Hat systems). Afterwards, run the following command to enable the startup of MySQL on system bootup:

```
shell> chkconfig --add mysql.server
```

No FreeBSD o script de inicialização normalmente deve ir no diretório '`/usr/local/etc/rc.d/`'. A página do manual `rc(8)` também diz que os scripts neste diretório só são executados, se o seu nome de base corresponder padrão global da sheel `*.sh`. Qualquer outro arquivo ou diretório presente dentro do diretório são silenciosamente ignorados. Em outra palavras, no FreeBSD você deve instalar o arquivo '`mysql.server`' como '`/usr/local/etc/rc.d/mysql.server.sh`' para habilitar a inicialização automática.

Como uma alternativa para o exposto acima, alguns sistemas operacionais também usam '`/etc/rc.local`' ou '`/etc/init.d/boot.local`' para inicializar serviços adicionais durante o boot. Para iniciar o MySQL usando este método, você poderia poderia adicionar algo como o seguinte a ele:

```
/bin/sh -c 'cd /usr/local/mysql; ./bin/mysqld_safe --user=mysql &'
```

Você também pode adicionar opções para `mysql.server` em um arquivo global '`/etc/my.cnf`'. Um típico arquivo '`/etc/my.cnf`' pode parecer com isto:

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql
```

```
[mysql.server]
basedir=/usr/local/mysql
```

O script `mysql.server` entende as seguintes opções: `datadir`, `basedir` e `pid-file`.

A seguinte tabela mostra quais grupos de opções cada script de inicialização lê dos arquivos de opções:

Script	**Grupos de opções**
`mysqld`	`[mysqld]`, `[server]` e `[mysqld-major-version]`
`mysql.server`	`[mysql.server]`, `[mysqld]`, e `[server]`
`mysqld_safe`	`[mysql.server]`, `[mysqld]`, e `[server]`

Para compatibilidade com versões anteriores, o `mysql.server` também lê o grupo `[mysql_server]` e `mysqld_safe` também lê o grupo `[safe_mysqld]`. No entanto, você deve atualizar os seus arquivos de opções para usar os grupos `[mysql.server]` e `[mysqld_safe]`.

Veja Seção 4.1.2 [Arquivos de Opções], Página 216.

2.5 Atualizando/Desatualizando o MySQL

Antes de fazer uma atualização, você deve fazer o backup de seus bancos de dados antigos.

Você sempre pode mover os arquivos de formato e de dados do MySQL entre diferentes versões na mesma arquitetura enquanto você tiver versão base do MySQL. A versão base atual é 4. Se você alterar o conjunto de caracteres quando executar o MySQL, você deve executar `myisamchk -r -q --set-character--set=charset` em todas tabelas. De outra forma seus índices podem não ser corretamente ordenados, porque alterar o conjunto de caracteres também pode alterar a ordenação.

Se você tem receio de novas versões, você sempre pode renomear seu antigo `mysqld` para algo como `mysqld`-'número-da-versão-antiga'. Se o seu novo `mysqld` comportar de maneira inesperada, você simplesmente pode desliga-lo e reiniciar com seu antigo `mysqld`!

Se depois de uma atualização, você tiver problemas com programas clientes recompilados como `Commands out of sync` ou "core dumps" inexperados, você provavelmente usou um arquivo de cabeçalho ou de biblioteca antigo na compilação de seus programas. Neste caso você deve conferir a data de seu arquivo '`mysql.h`' e da biblioteca '`libmysqlclient.a`' para verificar que eles são da nova distribuição MySQL. Se não, por favor, recompile seus programas!

Se você tiver problemas, como na inicialização do novo servidor `mysqld` ou caso você não consiga conectar sem uma senha, confira se o seu arquvo '`my.cnf`' é o mesmo da antiga instalação! Você pode conferir com isto: `nome-programa --print-defaults`. Se isto não produzir outra saída além do nome do programa, você tem um arquivo `my.cnf` ativo que está afetando a operacionalidade do servidor!

É uma boa idéia reconstruir e reinstalar o módulo `Perl DBD-mysql` sempre que instalar uma nova versão do MySQL. O mesmo se aplica para outras interfaces MySQL, como `Python MySQLdb`.

2.5.1 Atualizando da Versão 4.0 para 4.1

Varias comportamentos visíveis foram alteradas entre o MySQL 4.0 e o MySQL 4.1 para corrigir erros críticos e tornar o MySQL mais compatível com o padrão ANSI SQL. Estas alterações podem afetar à sua aplicação.

Alguns dos comportamentos do MySQL 4.1 no 4.0 podem ser testados antes de realizar uma atualização completa para a versão 4.1, adicionamos às últimas distribuições do MySQL 4.0 (a paritr da 4.0.12) a opção de inicialização `--new` para o `mysqld`.

Esta opção lhe dá o comportamento da versão 4.1 para as alterações mais críticas. Você também pode habilitar estes comportamentos para a conexão de uma determinado cliente com o comando `SET @@new=1`, ou desabilitá-lo se ele for iniciado com `SET @@new=0`.

Se você acredita que algumas das alterações da versão 4.1 o afetarão, recomendamos que antes de atualizar para a versão 4.1, você faça o download da última distribuição do MySQL 4.0 e o execute com a opção `--new` adicionando o seguinte ao seu arquivo de configuração:

```
[mysqld-4.0]
new
```

Deste modo você pode testar o novo comportamento com seus aplicativos na versão 4.0 para certificar-se que eles funcionam. Isto o ajudará a ter uma transição suave quando realizar uma atualização completa do MySQL 4.1. Fazendo isto do modo acima irá assegurar que você não execute acidentalemte a versão 4.1 com a opção `--new` mais tarde.

A seguinte lista descreve alterações que podem afetar aplicações e que você deve observar ao atualizar para a versão 4.1:

- `TIMESTAMP` agora é retornado como uma string com o formato `'YYYY-MM-DD HH:MM:SS'`. (A opção `--new` pode ser usada a partir da versão 4.0.12 para fazer um servidor 4.0 se comportar como 4.1 a este respeito.) Se você quiser tê-lo com um número (como a Versão 4.0 faz) deve-se adicionar `+0` a coluna `TIMESTAMP` a eles:

  ```
  mysql> SELECT ts_col + 0 FROM tbl_name;
  ```

  Tamanhos de display para `TIMESTAMP` não são mais suportados. Por exemplo, se você declarar um coluna como `TIMESTAMP(10)`, o `(10)` é ignorado.

  Esta mudança era necessária para compatibilidade com os padrões SQL. Em uma versão futura. Em uma versão futura, uma alteração adicional será feita (compatível com versões anteriores com esta mudaça), permitindo que o tamanho do timestamp indique o número desejado de dígitos de frações de um segundo.

- Valores binários (`0xFFDF`) agora são assumidos como strings em vez de números. Isto corrige o problema com conjunto de caracteres onde é conveniente colocar a string como um valor binário. Com esta alteração você deve usar `CAST()` se você quiser comparar valores binários numericamente como inteiros:

  ```
  SELECT CAST(0XFEFF AS UNSIGNED INTEGER) < CAST(0XFF AS UNSIGNED INTEGER)
  ```

  Se você não usa `CAST()`, uma comparação lexicográfica da string será feita:

  ```
  mysql> SELECT 0xFEFF < 0xFF;
          -> 1
  ```

  Usando itens binários em um contexto numérico ou comparando-os usando o operador `=` deve funcionar como antes. (A opção `--new` pode ser usado para fazer o servidor 4.0 se comportar como 4.1 a partir da versão 4.0.13.)

- Para funções que produzem um valor `DATE`, `DATETIME`, ou `TIME`, o resultado retornado para o cliente agora está corrigido para ter um tipo temporal. Por exemplo, no MySQL 4.1, você tem este resultado:

  ```
  mysql> SELECT CAST("2001-1-1" as DATETIME);
      -> '2001-01-01 00:00:00'
  ```

  No MySQL 4.0, o resultado é diferente:

  ```
  mysql> SELECT CAST("2001-1-1" as DATETIME);
  -> '2001-01-01'
  ```

- Valores `DEFAULT` não podem mais ser especificado para colunas `AUTO_INCREMENT` (Na versão 4.0, um valor `DEFAULT` é ignorado sem aviso, na 4.1 ocorre um erro).
- `LIMIT` não aceita mais argumentos negativos. Use 18446744073709551615 em vez de -1.
- `SERIALIZE` não é mais uma opção válida para `sql_mode`. Deve-se usar `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE`. `SERIALIZE` também não é mais válido para a opção `--sql-mode` do `mysqld`. Use `--transaction-isolation=SERIALIZABLE`
- Todas tabelas e colunas strings agora têm um conjunto de caracter. Veja Cap“ptexi tulo 9 [Charset], Página 710. A informação do conjunto de caracteres é mostrada por `SHOW CREATE TABLE` e `mysqldump`. (O MySQL versão 4.0.6 e acima pode ler o novo arquivo dump; versões mais antigas não podem.)
- O formato de definição de tabela usado nos arquivos '`.frm`' mudaram um pouco na versão 4.1. O MySQL 4.0.11 e adiante leêm o novo formato '`.frm`' diretamente, mas versões mais antigas não podem. Se você precisa mover tabelas da versão 4.1. para uma mais nova que a 4.0.11, você de usar `mysqldump`. Veja Seção 4.9.7 [`mysqldump`], Página 361.
- Se você estiver executando vários servidores na mesma máquina Windows, você deve usar uma opção `--shared_memory_base_name` diferentes para cada máquina
- A interface para agrupar funções UDF alterou um pouco. Você deve agora declarar uma função `xxx_clear()` para cada função de agrupamento.

Em geral, atualizar para o MySQL 4.1 a partir de uma versão mais nova do MySQL envolve os serguintes passos:

- Verifique na seção de alterações se houve alguma mudança que pode afetar a sua aplicação.
- Leia os novos itens da versão 4.1 para ver quais itens interessantes que você pode usar na versão 4.1. Veja Seção D.2 [Novidades na versão 4.1.x], Página 957.
- Se você estiver executando o MySQL Server no Windows, veja também Seção 2.5.8 [Windows upgrading], Página 132.
- Após o upgrade, atualize a tabela de permissões para gerar uma nova coluna `Password` maior que é necessária para tratamento seguro de senhas. O procedimento usa `mysql_fix_privilege_tables` e está descrito em Seção 2.5.6 [Upgrading-grant-tables], Página 130. Estratégias alternativas para tratamento de senhas depois de uma atualização estão descritos posteriormente nesta seção.

O mecanismo de hashing da senha foi alterado na versão 4.1 para fornecer maior segurança, mas ele pode causar problemas de compatibilidade se você ainda tiver clientes que usam a

biblioteca cliente 4.0 ou anterior. (É bastante indesejável que você tenha clientes 4.0 em situações onde o cliente conecta de uma máquina remota que ainda não tenha sido atualizada para a versão 4.1). A seguinte lista indica algumas estratégias possíveis de atualização. Elas representam o que se deve fazer para escolher se ter compatibilidade com clientes antigos e ter maior segurança.

- Não atualizar para a versão 4.1. Nenhum comportamento será alterado, mas é claro que você não poderá usar qualquer um dos novos recursos fornecido pelo protocolo cliente/servidor da versão 4.1. (O MySQL 4.1 tem um protocolo cliente/servidor extendido que oferece tais recursos como instruções preparadas e conjuntos de múltiplos resultados.) Veja Seção 12.1.4 [C API Prepared statements], Página 828.
- Atualizar para a versão 4.1 e executar o script `mysql_fix_privilege_tables` para aumentar a coluna `Password` na tabela `user` e assim poder guardar hashes de senhas longos. Mas execute o servidor com a opção `--old-passwords` para fornecer compatibilidade com versões anteriores que premitem que clientes pre-4.1 continuem a conectar em suas contas de hash curto. Eventualmente, quando todos os seus clientes estiverem atualizados para a versão 4.1, você pode parar de usar a opção do servidor `--old-passwords`. Você também pode alterar as senhas em sua conta MySQL para usar o novo formato que é mais seguro.
- Atualizar para versão 4.1 e executar o script `mysql_fix_privilege_tables` para aumentar a coluna `Password` na tabela `user`. Se você sabe que todos os clientes também foram atualizados para a versão 4.1, não execute o servidor com a opção `--old-passwords`. Em vez disso, altere a senha em todas as contas existentes para que elas tenham o novo formato. Uma instalação pura da versão 4.1 é o mais seguro.

Informações adicionais sobre hashing de senha em relação a autenticação no cliente e operações de alteração de senha podem ser encontrados em Seção 4.3.11 [Password hashing], Página 245.

2.5.2 Atualizando da Versão 3.23 para 4.0

Em geral, o que você deve fazer é atualizar para a versão 4.0 um versão mais nova do MySQL:

- Após o upgrade, atualize a tabela de permissões para adicionar novos privilégios e recursos. O procedimento usa o script `mysql_fix_privilege_tables` e está descrito em Seção 2.5.6 [Upgrading-grant-tables], Página 130.
- Edite qualquer script de inicialização ou arquivo de configuração para não utilizar nenhuma das opções obsoletas listadas posteriormente nesta seção.
- Converta seua arquivos `ISAM` antigos para arquivos `MyISAM` com o comando: `mysql_convert_table_format database`. (Este é um script Perl; ele exige que o DBI esteja instalado). Paa converter a tabela em um dado banco de dados, use este comando:

  ```
  shell> mysql_convert_table_format database db_name
  ```

  Note que ele deve ser usado apenas se você usar se todas as tabelas em um dado banco de dados são `ISAM` ou `MyISAM`. Para evitar a conversão de tabelas de outros tipos para `MyISAM`, você pode listar explicitamente o nome de suas tabelas `ISAM` depois do nome do banco de dados na linha de comando. Você também pode executar uma instrução

`ALTER TABLE table_name TYPE=MyISAM` para cada tabela `ISAM` para convertê-la para `MyISAM`.

Para descobir o tipo de uma determinada tabela, use esta instrução:

```
mysql> SHOW TABLE STATUS LIKE 'tbl_name';
```

- Certifique-se de que você não tem nenhum cliente MySQL que utiliza bibliotecas compartilhadas (com o Perl `DBD-mysql`). Se você tiver, você deve recompilá-las já que as estruturas usadas em '`libmysqlclient.so`' foram alteradas. O mesmo se aplica a outras interfaces MySQL, como Python `MySQLdb`.

O MySQL 4.0 funcionará mesmo se você não fizer o acima, mas você não poderá usar os novos privilégios de segurança pois o MySQL 4.0 e você podem encontrar problemas ao atualizar o MySQL para a versão 4.1 ou mais nova. O formato do arquivo `ISAM` ainda funciona no MySQL 4.0 mas está obsoleto e será disabilitado (não compilado por padrão) no MySQL 4.1. Em vez disso deve se usar tabelas `MyISAM`.

Clientes antigos devem funcionar com um servidor versão 4.0 sem nenhum problema.

Mesmo se você fizer o indicado acima, você ainda pode voltar para o MySQL 3.23.52 ou mais novo se você encontrar problemas com o MySQL da série 4.0. Neste caso você deve usar o `mysqldump` para fazer um dump de qualquer tabela que use um índice full-text e recarregar o arquivo de dump no servidor 3.23 (pois o 4.0 usa um novo formato para índices full-text).

A seguir está uma lista mais completa com o que deve ser observado para atualizar para a versão 4.0;

- O MySQL 4.0 tem vários novos privilégios na tabela `mysql.user`. Veja Seção 4.4.1 [GRANT], Página 254.

  Para fazer estes novos privilégios funcionarem, deve se atualizar a tabela de permissões. O procedimento está descrito em Seção 2.5.6 [Upgrading-grant-tables], Página 130. Até que este script esteja executando todos os usuários têm os privilégios `SHOW DATABASES`, `CREATE TEMPORARY TABLES` e `LOCK TABLES`. Os privilégios `SUPER` e `EXECUTE` tiram o seu valor de `PROCESS`. `REPLICATION SLAVE` e `REPLICATION CLIENT` tiram o seu valor de `FILE`.

  Se você tiver qualquer script que crie novos usuários, você pode querer alterá-los para usar os novos privilégios. Se você não está usando o comando `GRANT` nos scripts, este é um bom momento para alterar os seus scripts e usar `GRANT` em vez de modificar a tabela de permissões diretamente.

  A partir da versão 4.0.2 a opção `--safe-show-database` está obsoleta (e não faz mais nada). Veja Seção 4.3.3 [Opções de privilégio], Página 230.

  Se você receber um erro `Access denied` para novos usuários na versão 4.0.2, você deve verificar se você precisa de alguma das novas concessões que você não precisava antes. Em particular, você precisará `REPLICATION SLAVE` (em vez de `FILE`) para novos slaves.
- '`safe_mysqld`' é renomeado para '`mysqld_safe`'. Para compatibilidade com versões anteriores, as distribuições binárias, irão, por algum tempo, incluir `safe_mysqld` como um link simbólico para `mysqld_safe`.
- Suporte para InnoDB agora está incluído na distribuição binária. Se você contruir o MySQL a partir de um fonte, o InnoDB está configurado por padrão, Se você não usar o InnoDB e quiser economizar memória ao executar o servidor que possui suorte a

InnoDB habilitado, use a opção de inicialização do servidor. Para compilar o MySQL sem suporte ao InnoDB, execute `configure` com a opção `--without-innodb`.

- O parâmetro de inicialização `myisam_max_extra_sort_file_size` e `myisam_max_extra_sort_file_size` são dados agora em bytes. (eram dados em megabytes antes da versão 4.0.3).

  O lock de sistema externo dos arquivos MyISAM/ISAM agora está desligado por padrão. Pode se ligá-los fazendo `--external-locking`. (Para a maioria dos usuários isto nunca é necessário).

- A seguintes variáveis/opções de inicializaçao foram renomeadas:

Nome Antigo	**Novo Nome**.
`myisam_bulk_insert_tree_size`	`bulk_insert_buffer_size`
`query_cache_startup_type`	`query_cache_type`
`record_buffer`	`read_buffer_size`
`record_rnd_buffer`	`read_rnd_buffer_size`
`sort_buffer`	`sort_buffer_size`
`warnings`	`log-warnings`
`--err-log`	`--log-error` (para `mysqld_safe`)

  As opções de inicialização `record_buffer`, `sort_buffer` e `warnings` ainda funcionarão no MySQL 4.0 mas estãp obsoletas.

- As seguintes veriáveis SQL mudaram o nome.

Nome Antigo	**Novo Nome**.
`SQL_BIG_TABLES`	`BIG_TABLES`
`SQL_LOW_PRIORITY_UPDATES`	`LOW_PRIORITY_UPDATES`
`SQL_MAX_JOIN_SIZE`	`MAX_JOIN_SIZE`
`SQL_QUERY_CACHE_TYPE`	`QUERY_CACHE_TYPE`

  Os nomes antigos ainda funcionam no MySQL 4.0 mas estão obsoletos.

- Você deve usar `SET GLOBAL SQL_SLAVE_SKIP_COUNTER=#` em vez de `SET SQL_SLAVE_SKIP_COUNTER=#`.
- As opções de inicialização `--skip-locking` e `--enable-locking` foram renomeadas para `--skip-external-locking` e `--external-locking`.
- `SHOW MASTER STATUS` agora retorna um conjunto vazio se o log binário não estiver habilitado.
- `SHOW SLAVE STATUS` agora retorna um conjunto vazio se o slave não está inicializado.
- O `mysqld` agora tem a opção `--temp-pool` habilitada por padrão já que isto da melhor rendimento com alguns SO (Principalmente no Linux).
- Colunas `DOUBLE` e `FLOAT` agora respeitam o parâmetro `UNSIGNED` no armazenamento (antes, `UNSIGNED` era ignortado por estas colunas).
- `ORDER BY coluna DESC` ordena valores `NULL` por último, como no MySQL 4.0.11. Na versão 3.23 e anteriores da versão 4.0, isto nem sempre era consistente.
- `SHOW INDEX` tem duas colunas a mais (`Null` e `Index_type`) que ele tinha nas versões 3.23.
- `CHECK`, `SIGNED`, `LOCALTIME` e `LOCALTIMESTAMP` são agora palavras reservadas.

- O resultado de todos os operadores bitwise (`|`, `&`, `<<`, `>>` e `~`) agora são unsigned. Isto pode causar problemas se você estiver usando-as em um contexto onde você quer um resultado com sinal. Veja Seção 6.3.5 [Funções de Conversão], Página 544.
- **Nota**: quando você usa subtração entre valores inteiros onde um deles é do tipo `UNSIGNED`, o resultado será sem sinal. Em oyras palavras, antes de atualizar para o MySQL 4.0, você deve verificar sua aplicação para os casos onde você está subtraindo um valor de uma entidade sem sinal e quer um número negativo como resposta ou subtraindo um valor sem sinal de uma coluna do tipo inteiro. Você pode disabilitar este comportamento usando a opção `--sql-mode=NO_UNSIGNED_SUBTRACTION` ao iniciar o `mysqld`. Veja Seção 6.3.5 [Funções de conversão], Página 544.
- Para usar `MATCH ... AGAINST (... IN BOOLEAN MODE)` com suas tabelas, você precisa recontruí-las com `REPAIR TABLE nome_tabela USE_FRM`.
- `LOCATE()` e `INSTR()` são caso sensitivo se um dos argumentos é uma string binária. De outra forma elas são caso-insensitivo.
- `STRCMP()` agora usa o conjunto de caracteres atual ao fazer comparações, o que significa que o comportamento padrão das comparações agora é caso-insensitivo.
- `HEX(string)` agora retorna os caracteres na `string` convertidos para hexadecimal. Se você quiser converter um número para hexadecimal, você deve se assugurar que você chama `HEX()` com um argumento numérico.
- Na versão 3.23, `INSERT INTO ... SELECT` sempre tem o `IGNORE` habilitado. Na versão 4.0.1, o MySQL irá parar (e possívelmente fazer um roll back) por padrão no caso de '`mysqld_safe`' ser renomeado para '`mysqld_safe`'. Por algum tempo incluiremos em nossa distribuição binária o `mysqld_safe` como um link simbólico para `mysqld_safe`.
- um erro se você não especificar `IGNORE`.
- As funções antigas da API C `mysql_drop_db()`, `mysql_create_db()` e `mysql_connect()` não sã mais suportadas a menos que você compile o MySQL com `CFLAGS=-DUSE_OLD_FUNCTIONS`. No entanto, é preferível alterar o cliente para utilizar a nova API 4.0.
- Na estrutura `MYSQL_FIELD`, `length` e `max_length` foram alterados de `unsigned int` para `unsigned long`. Isto não deve causar problemas, exceto que eles podem gerar mensagens de avisos quando quando usado como argumento em uma classe `printf()` de funções.
- Você deve usar `TRUNCATE TABLE` quando quiser deletar todos os registros de uma tabela e você não precisa obter uma contagen de quantas colunas forma deletadas. (`DELETE FROM table_name` retorna a contagem de linhas na versão 4.0, e `TRUNCATE TABLE` é mais rápido.)
- Você receberá um erro se tiver um `LOCK TABLES` ativo ou transações ao tentar executar `TRUNCATE TABLE` ou `DROP DATABASE`.
- Você deve usar inteiros para armazenar valores em colunas `BIGINT` (em vez de usar strings, como você fez no MySQL 3.23). Usar strings ainda funicona, mas usar inteiros é mais eficiente.
- O formato de `SHOW OPEN TABLE` alterou.
- Clientes multi-thread devem usar `mysql_thread_init()` e `mysql_thread_end()`. Veja Seção 12.1.14 [Clientes em threads], Página 863.

- Se você quiser recompilar o módulo Perl `DBD::mysql`, você deve conseguir o `DBD-mysql` versão 1.2218 ou mais novo porque os módulos DBD mais antigos usam a chamada obsoleta `mysql_drop_db()`. A versão 2.1022 ou mais nova é recomendada.
- Na versão `RAND(seed)` retorna uma série de número randômicas diferente que na 3.23; isto foi feito para uma diferenciação maior de `RAND(seed)` e `RAND(seed+1)`.
- O tipo padrão retornado por `IFNULL(A,B)` agora está configurado para ser o mais 'geral' dos tipos de `A` e `B`. (A ordem geral-para-específco é string, `REAL` ou `INTEGER`).

Se você estiver executando o MySQL Server no Windows, veja Seção 2.5.8 [Atualizando o Windows], Página 132. Se você estiver usando replicação, veja Seção 4.11.2 [Replication Implementation], Página 379.

2.5.3 Atualizando da versão 3.22 para 3.23

A Versão 3.23 do MySQL suporta tabelas do novo tipo `MyISAM` e do antigo tipo `ISAM`. Você não necessita converter suas antigas tabelas para usá-las com a versão 3.23. Por padrão, todas novas tabelas serão criadas usando o tipo `MyISAM` (a menos que você inicie o `mysqld` com a opção `--default-table-type=isam`). Você pode converterr uma tabela `ISAM` para uma formato `MyISAM` com `ALTER TABLE nome_tabela TYPE=MyISAM` ou com o script Perl `mysql_convert_table_format`.

Os clientes versões 3.22 e 3.21 irão trabalhar sem quaisquer problemas com um servidor versão 3.23.

As seguintes listas dizem o que você deve conferir quando atualizar para a versão 3.23:

- Todas tabelas que usam o conjunto de caracteres `tis620` devem ser corrigidos com `myisamchk -r` ou `REPAIR TABLE`.
- Se você fizer um `DROP DATABASE` em um banco de dados ligado simbolicamente, a ligação e o banco de dados original serão apagados. (Isto não acontece na 3.22 porque o `configure` não detecta a disponibilidade da chamada de sistema `readlink`).
- `OPTIMIZE TABLE` agora funciona somente para tabelas **MyISAM**. Para outros tipos de tabelas, você pode usar `ALTER TABLE` para otimizar a tabela. Durante o `OPTIMIZE TABLE` a tabela é, agora, bloqueada para prevenir que seja usada por outras threads.
- O cliente MySQL `mysql` é, agora, inicializado por padrão com a opção `--no-named-commands (-g)`. Esta opção pode ser desabilitada com `--enable-named-commands (-G)`. Isto pode causar problemas de imcompatibilidade em alguns casos, por exemplo, em scripts SQL que usam comandos sem ponto e vírgula! Comandos longos continuam funcionando.
- Funções de data que funcionam em partes de datas (como `MONTH()`) não retornará 0 para datas `0000-00-00`. (No MySQL 3.22 estas funções retornam `NULL`.)
- Se você estiver usando a ordem de classificação de caracteres `alemã` para tabelas `ISAM`, você deve reparar todas suas tabelas com `isamchk -r`, porque foram feitas alterações na sua ordem de classificação!
- O tipo padrão de retorno de `IF()` irá agora depender de ambos argumentos e não apenas do primeiro argumento.
- Colunas `AUTO_INCREMENT` não devem ser usadas para armazenar números negativos. A razão para isto é que números negativos causam problemas quando o -1 passa para 0.

Você não deve armazenar 0 em uma coluna AUTO_INCREMENT também; CHECK TABLE irá reclamar sobre valores 0 porque eles podem alterar se você fizer um dump e restaurar a tabela. AUTO_INCREMENT é, agora, tratado em um nível mais baixo para tabelas MyISAM e é muito mais rápido que antes. Para tabelas MyISAM números antigos também não são mais reusados, mesmo se você apagar algumas linhas da tabela.

- CASE, DELAYED, ELSE, END, FULLTEXT, INNER, RIGHT, THEN e WHEN agora são palavras reservadas.
- FLOAT(X) agora é um tipo de ponto flutuante verdadeiro e não um valor com um número fixo de decimais.
- Quando estiver declarando colunas usando o tipo DECIMAL(tamanho,dec, o argumento tamanho não inclui mais um lugar para o símbolo do ponto decimal.
- Uma string TIME agora deve estar em um dos seguintes formatos: [[[DAYS] [H]H:]MM:]SS[.fraction] ou [[[[[H]H]H]H]MM]SS[.fraction]
- LIKE agora compara strings usando as mesmas regras de comparação de caracteres que o operador '='. Se você precisa do antigo compartamento, você pdoe compilar o MySQL com a opção CXXFLGAS=-DLIKE_CMP_TOUPPER.
- REGEXP agora é caso insensitivo se nenhuma das strings forem binárias.
- Quando for necessário dar manutenção ou reparar tabelas MyISAM '.MYI' deve ser usado a instrução CHECK TABLE ou o comando myisamchk. Para tabelas ISAM (.ISM), use o comando isamchk
- Se desejar que os arquivos mysqldump sejam compatíveis entre as versões 3.22 e 3.23 do MySQL, não deve ser usados as opções --opt ou --full com o mysqldump.
- Confira todas suas chamadas à DATE_FORMAT() para ter certeza que exista um '%' antes de cada caractere formatador. (Versões mais antigas que o MySQL 3.22 aceitaivam esta sintaxe.)
- mysql_fetch_fields_direct() agora é uma função (era uma macro) e ela retorna um ponteiro para um MYSQL_FIELD no lugar de um MYSQL_FIELD.
- mysql_num_fields() não pode mais ser usada em um objeto MYSQL* (agora é uma função que obtem valores MYSQL_RES* como um argumento). Com um objeto MYSQL* agora voce deve usar mysql_field_count().
- No MySQL Versão 3.22, a saída de SELECT DISTINCT ... era na maioria das vezes ordenada. Na Versão 3.23, você deve usar GROUP BY ou ORDER BY para obter a saída ordenada.
- SUM() agora retorna NULL, em vez de 0 se não existir registros coincidentes. Isto é de acordo com o ANSI SQL.
- Um AND ou OR com valores NULL agora retornam NULL no lugar de 0. Isto afetará, em grande parte, pesquisas que usam NOT em uma expressão AND/OR como NOT NULL = NULL.
- LPAD() e RPAD() reduzirão a string resultante se ela for maior que o tamanho do argumento.

2.5.4 Atualizando da versão 3.21 para 3.22

Nada que afetaria a compatibilidade foi alterada entre a versão 3.21 e 3.22. A única dificuldade é que novas tabelas que são criadas com colunas do tipo `DATE` usarão a nova forma de armazenar a data. Você não pode acessar esses novos campos com uma versão antiga de `mysqld`.

Depois de instalar o MySQL versão 3.22, você deve iniciar o novo servidor e depois executar o script `mysql_fix_privilege_tables`. Isto adicionará os novos privilégios que você precisará para usar o comando `GRANT`. Se você se esquecer disto, sera retornado o erro `Access denied` quando você tentar usar `ALTER TABLE`, `CREATE INDEX` ou `DROP INDEX`. O procedimento para atualizar a tabela de permissões está descrito em Seção 2.5.6 [Upgrading-grant-tables], Página 130.

A interface API C para `mysql_real_connect()` foi alterada. Se você tem um programa cliente antigo que chama essa função, você deve colocar um `0` para o novo argumento `db` (ou recodificar o cliente para enviar o elemento `db` para conexões mais rápidas). Você também deve chamar `mysql_init()` antes de chamar `mysql_real_connect()`! Esta alteração foi feita para permitir à nova função `mysql_options()` salvar opções na estrutura do manipulador do `MYSQL`.

A variável `key_buffer` do `mysqld` foi renomeada para `key_buffer_size`, mas você ainda pode usar o antigo nome nos seus arquivos de inicialização.

2.5.5 Atualizando da versão 3.20 para 3.21

Se você estiver executando uma versão mais antiga que a Versão 3.20.28 e deseja mudar para a versão 3.21 você deve fazer o seguinte:

Inicie o servidor `mysqld` versão 3.21 com a opção `--old-protocol` para usá-lo com clientes de uma distribuição da versão 3.20 Neste caso, a nova função cliente `mysql_errno()` não irá retornar erro do servidor, somente `CR_UNKNOWN_ERROR` (mas isto funciona para erros de clientes) e o servidor usa a forma função `password()` anterior a 3.21 para verificação, ao invés do novo método.

Se você **NÃO** estiver usando a opção `--old-protocol` para `mysqld`, você precisará fazer as seguir alterações:

- Todo o código cliente deve ser recompilado. Se você usa o ODBC, deve obter o novo driver **MyODBC** 2.x.
- O script `scripts/add_long_password` deve ser executado para converter o campo `Password` na tabela `mysql.user` para `CHAR(16)`.
- Todas as senhas devem ser reatribuidas na tabela `mysql.user` (para obter 62-bits no lugar de senhas 31-bits).
- O formato das tabelas não foi alterado, então não é preciso converter nenhuma tabela.

A versão do MySQL 3.20.28 e superiores podem manipular o novo formato da tabela de `usuários` sem afetar os clientes. Se você tem uma versão do MySQL mais nova que 3.20.28, senhas não irão mais funcionar se você converter a tabela de `usuaios`. Por segurança, você primeiro deve fazer uma atualização para a versão 3.20.28, pelo menos, e então atualizar para a versão 3.21.

O novo código cliente trabalha com um servidor `mysqld` 3.20.x, portanto se houver problemas com 3.21.x você deve usar o antigo servidor 3.20.x sem a necessidade de recompilar os clientes novamente.

Se você não está usando a opção `--old-protocol` para o `mysqld`, antigos clientes não poderão se conectar e exibirão a seguinte mensagem de erro:

```
ERROR: Protocol mismatch. Server Version = 10 Client Version = 9
```

A nova interface PERL `DBI/DBD` também suporta a antiga interface `mysqlperl`. A única alteração que deve ser feita se você usa o `mysqlperl` é alterar os argumentos para a função `connect()`. Os novos argumentos são: `host`, `database`, `user`, `password` (note que os argumentos `user` e `password` foram alterados de lugar). Veja Seção 12.5.2 [Perl `DBI` Class], Página 881.

As seguintes alterações podem afetar consultas em antigas aplicações:

- `HAVING` deve ser especificada antes de qualquer cláusula `ORDER BY`.
- Os parâmetros para `LOCATE()` foram trocados.
- Agora existem algumas palavras reservadasi novas. As mais notáveis são `DATE TIME` e `TIMESTAMP`.

2.5.6 Atualizando a Tabela de Permissões

Algumas distribuições introduzem alterações a estrutura da tabelas de permissões (a tabela no banco de dados `mysql`) para adicionar novos privilégios ou recursos. Para ter certeza de que as suas tabelas de permissões estão corretas quando você atualizar para uma nova versão do MySQL, você deve atualizar a sua tabela de permissão também.

Em sistemas Unix ou semelhantes, atualize a tabela de permissões executando o script `mysql_fix_privilege_tables`:

```
shell> mysql_fix_privilege_tables
```

Você deve executar este script enquanto o servidor está em execução. Ele tenta se conectar ao servidor na máquina local como `root`. Se sua conta `root` exige uma senha, indique a senha na linha de comando. Para o MySQL 4.1 e acima, especifique a senha assim:

```
shell> mysql_fix_privilege_tables --password=senha_root
```

Antes do MySQL 4.1, especifique a senha desta forma:

```
shell> mysql_fix_privilege_tables senha_root
```

O script realiza `mysql_fix_privilege_tables` qualquer ação necessária para converter sua tabela de permissões para o formato atual. Você pode ver alguns avisos `Duplicate column name`, durante a execução, eles podem ser ignorados.

Depois de executar o script, pare o servidor e o reinicie.

No Windows, não existe uma modo fácil de se atualizar a tabela de permissões até o MySQL 4.0.15. A partir desta versão, as distribuições do MySQL incluem um script SQL `mysql_fix_privilege_tables.sql` que você pode executar usando o cliente `mysql`. Se sua instalação do MySQL está localizada em '`C:\mysql`', o comando se parecerá com este:

```
C:\mysql\bin> mysql -u root -p mysql

mysql> SOURCE C:\mysql\scripts\mysql_fix_privilege_tables.sql
```

Se sua instalação está localizada em algum outro diretório, ajuste o caminha apropriadamente.

O comando irá lhe pedir a senha do `root`; digite-a quando pedido.

Como no procedimento com o Unix, você pode ver alguns avisos `Duplicate column name` enquanto o `mysql` processa as instruções no script `mysql_fix_privilege_tables.sql`; eles podem ser ignorados.

Depois de executar o script, para o servidor e reinicie-o.

2.5.7 Atualizando para outra arquitetura

Se você estiver usando o MySQL Versão 3.23, você pode copiar os arquivos `.frm`, `.MYI` e `.MYD` para tabelas `MyISAM` entre diferentes arquiteturas que suportem o mesmo formato de ponto flutuante. (O MySQL cuida de cada detalhe de troca de bytes.) Veja Seção 7.1 [`MyISAM` Tables], Página 631.

Os arquivos `ISAM` de dados e índices ('`*.ISD`' e '`*.ISM`' respectivamente) são dependentes da arquitetura e em alguns casos dependentees do Sistema Operacional. Se você deseja mover suas aplicações para outra máquina que tem uma arquitetura ou SO diferentes da sua máquina atual, você não deve tentar mover um banco de dados simplesmente copiando os arquivos para a outra máquina. Use o `mysqldump`.

Por padrão, o `mysqldump` irá criar um arquivo contendo declarações SQL. Você pode então transferir o arquivo para a outra máquina e alimentá-la como uma entrada para o cliente `mysql`.

Utilize `mysqldump --help` para ver quais opções estão disponíveis. Se você está movendo os dados para uma versão mais nova do MySQL, você deve usar `mysqldump --opt` com a nova versão para obter uma descarga rápida e compacta.

A mais fácil (mas não a mais rápida) forma para mover um banco de dados entre duas máquinas é executar os seguintes comandos na máquina em que o banco de dados se encontra:

```
shell> mysqladmin -h 'nome da outra maquina' create nome_bd
shell> mysqldump --opt nome_bd \
        | mysql -h 'nome da outra maquina' nome_bd
```

Se você deseja copiar um banco de dados de um máquina remota sobre uma rede lenta, pode ser usado:

```
shell> mysqladmin create nome_bd
shell> mysqldump -h 'nome de outra maquina' --opt --compress nome_bd \
        | mysql nome_bd
```

O resultado pode também ser armazenado em um arquivo, depois transfira o arquivo para a máquina destino e carregue o arquivo no banco de dados. Por exemplo você pode descarregar um banco de dados para um arquivo na máquina origem desta forma:

```
shell> mysqldump --quick nome_bd | gzip > nome_bd.contents.gz
```

(O arquivo criado neste exemplo está compactado.) Transfria o arquivo contendo o conteúdo do banco de dados para a máquina destino e execute estes comandos:

```
shell> mysqladmin create nome_bd
shell> gunzip < nome_bd.contents.gz | mysql nome_bd
```

Também pode ser usado `mysqldump` e `mysqlimport` para ajudar na transferência do banco de dados. Para grandes tabelas, isto é muito mais rápido do que usar simplesmente `mysqldump`. Nos comandos abaixo, `DUMPDIR` representa o caminho completo do diretório que você utiliza para armazenar a saída de `mysqldump`.

Primeiro, crie o diretório para os arquivos de saída e descarregue o banco de dados:

```
shell> mkdir DUMPDIR
shell> mysqldump --tab=DUMPDIR nome_bd
```

Depois transfira os arquivo no diretório `DUMPDIR` para algum diretório correspondente na máquina destino e carregue os arquivos no MySQL assim:

```
shell> mysqladmin create nome_bd          # cria o banco de dados
shell> cat DUMPDIR/*.sql | mysql nome_bd  # cria tabelas no banco de dados
shell> mysqlimport nome_bd DUMPDIR/*.txt  # carrega dados nas tabelas
```

Não se esqueça de copiar o banco de dados `mysql` também, porque é nele que as tabelas de permissões (`user`, `db` e `host`) são armazenadas. Você pode ter que executar comandos como o usuário `root` do MySQL na nova máquina até que você tenha o banco de dados `mysql` no lugar.

Depois de importar o banco de dados `mysql` para a nova máquina, execute `mysqladmin flush-privileges` para que o servidor recarregue as informações das tabelas de permissões.

2.5.8 Atualizando o MySQL no Windows

Qaundo atualizar o MySQL no Windows, siga os passo abaixo:

1. Faça o download do última distribuição MySQL do Windows.
2. Escolha uma hora do dia com pouco uso, onde a parada para manutenção é aceitável.
3. Alerte os usuários que ainda estão ativos para sua parada de manutenção.
4. Pare o Servidor MySQL em execução (por exemplo, com `NET STOP mysql` ou com o utilitário de `Serviços` se você estiver exeutando MySQL como um serviço, ou com `mysqladmin shutdown`).
5. Finalize o programa `WinMySQLAdmin` se ele estiver em execução.
6. Execute o script de instalação do arquivo de distribuição do Windows, clicando no botão "Install" no WinZip e seguindo os passos da instalação do script.
7. Você pode sobrescrever a sua instalação antiga do MySQL (normalmente em '`C:\mysql`'), ou instalá-la em um diretório diferente, como `C:\mysql4`. Sobrescrever a instalação antiga é o recomendado.
8. Reinicie o serviço MySQL Server (por exemplo, com `NET START mysql` se você executar o MySQL como um serviço, ou chamado o `mysqld` diretamente).
9. Atualize a tabela de permissões. O procedimento está descrito em Seção 2.5.6 [Upgrading-grant-tables], Página 130.

Situações de erros possíveis:

```
A system error has occurred.
System error 1067 has occurred.
The process terminated unexpectedly.
```

Este erro significa que seu arquivo '`my.cnf`' (por padrão '`C:\my.cnf`') contém uma opção que não pode ser reconhecido pela MySQL. Você pode verificar que este é o caso tentando reiniciar o MySQL com o arquivo '`my.cnf`' renomeado, por exemplo, para '`my_cnf.old`' para prevenirt o servidor de usá-lo. Uma vez verificado isto, você precisa identificar qual parâmetro é o culpado. Crie um novo arquivo '`my.cnf`' e mova as partes do arquivo antigo para ele (reiniciando o servidor depois de mover cada parte) até que você determine qual opção está fazendo a inicialização do servidor falhar.

2.6 Notas específicas para os Sistemas Operacionais

2.6.1 Notas Windows

Esta seção descreve assuntos específicos para usar MySQL no Windows.

2.6.1.1 Conectando em um MySQL Rematamente a Windows Utilizando SSH

Aqui temos notas sobre como conectar a um servidor MySQL através de uma conexão remota e segura usando o SSH (por David Carlson `dcarlson@mplcomm.com`:

1. Instale um cliente SSH na sua máquina Windows. Como um usuário, o melhor opção paga que encontrei é o `SecureCRT` da `http://www.vandyke.com/`. Outra opção é o `f-secure` da `http://www.f-secure.com/`. Você também pode encontrar algumas versões livres no **Google** em `http://directory.google.com/Top/Computers/Security/Products_and_Tools/Cryptography/SSH/Clients/Windows/`.
2. Inicie seu cliente SSH Windows. Configure `Host_Name = IP_ou_Nome_servidormysql`. Configure `userid=seu_userid` para logar no seu servidor. Este valor `userid` não pode ser o mesmo do nome do usuário se sua conta MySQL.
3. Configure a porta de acesso. E também faça um acesso remoto (Configure `local_port: 3306`, `remote_host: ip_ou_nomeservidormysql`, `remote_port: 3306`) ou um acesso local (configure `port: 3306`, `host: localhost`, `remote port: 3306`).
4. Salve tudo, senão você terá que refazer tudo da próxima vez.
5. Logue ao seu servidor com a sessão SSH que acabou de ser criada.
6. Na sua máquina Windows, inicie algumas aplicações ODBC (como o Access).
7. Crie um novo arquivo no Windows e ligue ao MySQL usando o driver ODBC da mesma forma que você normalmente faz, EXCETO pelo fato de digitar `localhost` para a máquina servidora MySQL — não `nomeservidormysql`.

Você agora deve ter uma conexão ODBC ao MySQL, criptografada com SSH.

2.6.1.2 Compilando clientes MySQL no Windows

Em seus arquivos fontes, você deve incluir '`my_global.h`' antes de '`mysql.h`':

```
#include <my_global.h>
#include <mysql.h>
```

'`my_global.h`' inclui qualquer outro arquivo necessário para compatibilidade de Windows (como o '`windows.h`') se o arquivo é compilado no Windows.

Você também pode ligar seu código coma biblioteca dinâmica '`libmysq.lib`', que é apenas um wrapper para carregar em '`libmysql.dll`' sobre demanda, ou ligar com a biblioteca estática '`mysqlclient.lib`'.

Perceba que como as bibliotecas clientes do MySQL são compiladas como bibliotecas threaded, você também deve compilar seu código para ser multi-threaded!

2.6.1.3 MySQL para Windows Comparado com o MySQL para Unix

O MySQL para Windows tem provado ser muito estável. Esta versão do MySQL tem os mesmos recursos que sua versão correspondente Unix com as seguintes exceções:

Win95 e threads

O Win95 perde aproximadamente 200 bytes de memória principal para cada thread criada. Cada conexão no MySQL cria uma nova thread, portanto você não deve executar o `mysqld` por um longo tempo no Win95 se seu servidor lida com várias conexões! WinNT e Win98 não sofrem deste bug.

Leituras simultâneas

O MySQL depende das chamadas `pread()` e `pwrite()` para estar apto a misturar `INSERT` e `SELECT`. Atualmente nós usamos mutexes para emular `pread()`/`pwrite()`. Nós iremos, a longo prazo, trocar o nível da interface de arquivos com uma interface virtual para que nós possamos usar a interface `readfile()`/`writefile()` no NT/2000/XP para obter mais velocidade. A implementação atual limita o número de arquivos abertos que o MySQL pode usar para 1024, o que significa que você não conseguirá executar tantas threads simultâneas no NT/2000/XP como no Unix.

Leitura de blocos

O MySQL usa uma leitura de blocos para cada conexão, que tem as seguintes implicações:

- Uma conexão não irá ser disconectada automaticamente depois de 8 horas, como acontece com a versão Unix do MySQL.
- Se uma conexão trava, é impossível a finaliza-la sem matar o MySQL.
- `mysqladmin kill` não irá funcionar em uma conexão adormecida.
- `mysqladmin shutdown` não pode abortar enquanto existirem conexões adormecidas.

Planejamos corrigir este problema quando nossos desenvolvedores Windows tiverem conseguido um boa solução.

`DROP DATABASE`
: Você não pode remover um banco de dados que está em uso por alguma thread.

Matando o MySQL do gerenciador de tarefas
: Você não pode matar o MySQL do gerenciador de tarefas ou com o utilitário shutdown no Win95. Você deve desligá-lo com `mysqladmin shutdown`.

Nomes case-insensitivo
: Nomes de arquivos não são caso sensitivo no Windows, portanto, nomes de bancos de dados e tabelas do MySQL também não são caso sensitivo no Windows. A única restrição é que os nomes de bancos de dados e tabelas devem usar o mesmo caso em uma sentença fornecida. Veja Seção 6.1.3 [Name case sensitivity], Página 473.

O caracter de diretório '\'
: Componentes de nomes de caminho no Win95 são separados pelo caracter '\' o qual também é o caractere de escape no MySQL. Se você estiver usando `LOAD DATA INFILE` ou `SELECT ... INTO OUTFILE`, use nomes de arquivo no estilo Unix com caracteres '/':

```
mysql> LOAD DATA INFILE "C:/tmp/skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:/tmp/skr.txt' FROM skr;
```

Uma alternativa é dobrar o caracter '/':

```
mysql> LOAD DATA INFILE "C:\\tmp\\skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:\\tmp\\skr.txt' FROM skr;
```

Problems with pipes.
: Pipes não funcionam com confiança na linha de comando do Windows. Se o pipe incluir o caracter `^Z` / `CHAR(24)`, o Windows achará que ele encontrou o fim de um arquivo e abortará o programa.

Isto é um problma principalmente quando se tenta aplicar um log binário como a seguir:

```
mysqlbinlog binary-log-name | mysql --user=root
```

Se você obter um problema aplicando o log e suspeitar que seja devido a um caracter `^Z`/`CHAR(24)` você pode usar a seguinte alternativa:

```
mysqlbinlog binary-log-file --result-file=/tmp/bin.sql
mysql --user=root --eexecute "source /tmp/bin.sql"
```

O último comando pode também ser usado para leitura em qualquer arquivo sql que contenha dados binários.

erro: `Can't open named pipe`
: Se você utiliza um servidor MySQL versão 3.22 no NT com o os programas clientes MySQL mais novos, será apresentado o seguinte erro:

```
error 2017: can't open named pipe to host: . pipe...
```

Isto ocorre porque a versão do MySQL usa named pipes no NT por padrão. Você pode evitar este erro usando a opção `--host=localhost` para os novos clientes MySQL ou criar um arquivo de opções '`c:\my.cnf`' que contenha a seguinte informação:

```
[client]
host = localhost
```

A partir da versão 3.23.50, named pipes são habilitados somente se o `mysqld-nt` ou `mysqld-nt-max` for iniciado com a opção `--enable-name-pipe`.

Erro `Access denied for user`

Se você tenta executar um programa cliente MySQL para conectar a um servidor em execução na mesma máquina, nas obtem o erro `Access denied for user: 'some-user@unknown' to database 'mysql'` quando acessar um servidor MySQL na mesma máquina, signifca que o MySQL não pode resolver seu nome de máquina corretamente.

Para corrigir isto, você deve criar um arquivo '`\Windows\hosts`' com a seguinte informação:

```
127.0.0.1       localhost
```

`ALTER TABLE`

Enquanto você está executando uma instrução `ALTER TABLE`, a tabela está bloqueada para ser usado por outras threads. Isto ocorre devido ao fato de que no Windows, você não pode deletar um aruivo que está em uso por outra threads. No futuro, podemos encontrar algum modo de contornarmos este problema.

`DROP TABLE`

`DROP TABLE` em uma tabela que está em uso por uma tabela `MERGE` não funcionará no Windows porque o manipulador do `MERGE` faz o mapeamento da tabela escondido da camada superior do MySQL. Como o Windows não permite que você delete arquivos que estão abertos, você primeiro deve descarregar todas as tabelas `MERGE` (com `FLUSH TABLES`) ou apagar a tabela `MERGE` antes de deletar a tabela. Corrigiremos isto assim que introduzirmos views.

`DATA DIRECTORY` **e** `INDEX DIRECTORY`

As opções `DATA DIRECTORY` e `INDEX DIRECTORY` para `CREATE TABLE` são ignoradas no Windows, porque ele não suporta links simbólicos.

Aqui estão alguns assuntos em aberto para qualquer um que queira melhorar o MySQL no Windows:

- Adicionar alguns ícones agradáveis para o start e shutdown na instalação do MySQL.
- Seria muito interessante conseguir matar o `mysqld` do gerenciador de tarefas. Para o momento, deve ser usado o `mysqladmin shutdown`.
- Portar o `readline` para Windows para uso na ferramenta de linha de comando `mysql`.
- Versões GUI dos clientes MySQL padrões (`mysql`, `mysqlshow`, `mysqladmin` e `mysqldump`) seria ótimo.
- Seria muito bom se as funções de leitura e escrita no socket em '`net.c`' fosse interrompíveis. Isto tornaria possível matar threads abertas com `mysqladmin kill` no Windows.
- Adicionar macros para usar os métodos mais rápidos de incremento/decremento de threads seguras fornecidos pelo Windows.

2.6.2 Notas Linux (Todas as versões)

As notas abaixo a respeito da **glibc** aplicam-se somente na situação quando o MySQL é construido por você mesmo. Se você está executando Linux em uma máquina x86, na maioria dos casos é muito melhor para você usar nosso binário. Nós ligamos nossos binários com a melhor versão alterada da **glibc**, podemos escolher as melhores opções do compilador, em uma tentativa de torná-la funcional para um servidor muito exigido. Para um usuário comum, mesmo para configurações com várias conexões concorrentes e/ou tabelas excedendo o limite de 2 GB, nosso binário é, na maioria das vezes, a melhor escolha. Portanto se você ler o texto abaixo, e está em dúvida sobre o que deve fazer, tente usar o nosso binário primeiro para ver se ele preenche suas necessidades, e preocupe-se com uma construção própria apenas se você descobrir que nosso binário não é bom o suficiente para você. Neste caso, iríamos apreciar se fosse feito uma observação sobre isto, para que possamos fazer uma melhor versão binária da próxima vez.

O MySQL usa LinuxThreads no Linux. Se você usa uma versão do Linux que não tenha a `glibc2`, você deve instalar LinuxThreads antes de tentar compilar o MySQL. Você pode obter o LinuxThreads em `http://www.mysql.com/downloads/os-linux.html`.

NOTA: Temos visto alguns problemas estranhos com o Linux 2.2.14 e MySQL em sistemas SMP; Se você tem um sistema SMP, recomendamos a atualização para o Linux 2.4! Seu sistema ficará mais rápido e mais estável.

Perceba que as versões da `glibc` iguais ou anteriores à Versão 2.1.1 tem um bug fatal no tratamento do `pthread_mutex_timedwait`, que é usado quando você executar instruções `INSERT DELAYED`. Recomendamos não usar `INSERT DELAYED` antes de atualizar a `glibc`.

Se você planeja ter mais de 1000 conexões simultâneas, será necessário fazer algumas alterações na LinuxThreads, recompile-a e religue o MySQL ao novo '`libpthread.a`'. Aumente `PTHREAD_THREADS_MAX` em '`sysdeps/unix/sysv/linux/bits/local_lim.h`' para 4096 e abaixe o `STACK_SIZE` no '`linuxthreads/internals.h`' para 256KB. Os caminhos são relativos à raiz da `glibc`. Note que o MySQL não será estável com cerca de 600-1000 conexões se o valor de `STACK_SIZE` for o padrão de 2MB.

Se você tiver um problema com o MySQL, no qual ele não consiga abrir vários arquivos ou conexões, pode ser que você não tenha configurado o Linux para lidar com o número de arquivos suficiente.

No Linux 2.2 e posteriores, você pode conferir o valor para a alocação dos arquivos fazendo:

```
cat /proc/sys/fs/file-max
cat /proc/sys/fs/dquot-max
cat /proc/sys/fs/super-max
```

Se você possui mais de 16M de memória, deve ser adicionado o seguinte no seu script de boot (ex. '`/etc/rc/boot.local`' no SuSE Linux):

```
echo 65536 > /proc/sys/fs/file-max
echo 8192 > /proc/sys/fs/dquot-max
echo 1024 > /proc/sys/fs/super-max
```

Você também pode executar os comandos acima da linha de comando como root, mas neste caso, os antigos limites voltarão a ser usados na próxima vez que o computador for reiniciado.

De forma alternativa, você pode configurar estes parâmteros durante a inicialização usando a ferramenta `sysctl`, que é usada por muitas distribuições Linux (No SuSE a partir da versão 8.0). Apenas grave os seguintes valores em um arquivo chamado '`/etc/sysctl.conf`':

```
# Aumente alguns valores para o MySQL
fs.file-max = 65536
fs.dquot-max = 8192
fs.super-max = 1024
```

You should also add the following to '`/etc/my.cnf`':

```
[mysqld_safe]
open-files-limit=8192
```

Os parâmetros acima permitem o MySQL criar até 8192 conexões + arquivos.

A constante `STACK_SIZE` na LinuxThreads controla o espaçamento das pilhas threads no espaço de endereçamento. Ela necessita ser grande o bastante para que tenha espaço o suficiente para a pilha de cada thread, mas pequena o bastante para manter a pilha de alguma thread executando dos dados globais `mysqld`. Infelizmente, a implementação Linux de `mmap()`, como descobrimos em experiências, irá desmapear uma região já mapeada se você solicitar o mapeamento de um endereço já em uso, zerando os dados de toda a página ao invés de retoernar. um erro. Portanto a segurança do `mysqld` ou qualquer outra aplicação baseada em threads depende do comportamento gentil do código que cria as threads. O usuário deve tomar medidas para certirficar-se que o número de threads em funcionamento em qualquer hora seja suficientemente baixo para que as pilhas das threads permaneçam longe do monte global. Com `mysqld` você deve reforçar este comportamento "gentil" configurando um valor razoável para a variável `max_connections`.

Se você mesmo construiu o MySQL e não deseja confusões corrigindo LinuxThreads, você deve configurar `max_connections` para um valor máximo de 500. Ele ainda deve ser menor se você tiver uma chave grande para o buffer, grandes tabelas heap, ou outras coisas que fazem o `mysqld` alocar muita memória ou se você estiver executando um kernel 2.2 com o patch de 2GB. Se você estiver usando nosso binário ou RPM versão 3.23.25 ou posterior, você pode seguramente configurar `max_connections` para 1500, assumindo que não há uma grande chave de buffer ou tabelas heap com grande quantidade de dados. Quanto mais você reduz `STACK_SIZE` em LinuxThreads mais threads você pode criar seguramente. Recomendamos os valores entre 128K e 256K.

Se você usa várias conexões simultâneas, você pode sofrer com um "recurso" do kernel 2.2 que penaliza um processo por bifurcar-se ou clonar um filho na tentativa de prevenir um ataque de separação. Isto faz com que o MySQL não consiga fazer uma bom escalonamento, quando o número de clientes simultâneos cresce. Em sistemas com CPU única, temos visto isto se manifestar em uma criação muito lenta das threads, tornando a conexão ao MySQL muito lenta. Em sistemas de múltiplas CPUs, temos observado uma queda gradual na velocidade das consultas quando o número de clientes aumenta. No processo de tentar encontrar uma solução, recebemos um patch do kernel de um de nossos usuários, que alega fazer muita diferença para seu site. O patch está disponível aqui (`http://www.mysql.com/Downloads/Patches/linux-fork.patch`). Atualmente temos feito testes extensivos deste patch nos sistemas de desenvolvimento e produção. A performance do `MySQL` obtem uma melhora significativa, sem causar problemas e atualmente o recomendamos para nossos usuários que continuando trabalhando com

servidores muito carregados em kernels 2.2. Este detalhe foi corrigido no kernel 2.4, portanto, se você não está satisfeito com a performance atual do seu sistema, melhor do que aplicar um patch ao seu kernel 2.2, pode ser mais fácil simplesmente atualizar para o 2.4, que lhe dará também uma melhora em seu sistemas SMP em adição à correção do bug discutido aqui.

Estamos testando o MySQL no kernel 2.4 em uma máquina com 2 processadores e descobrimos que o MySQL escalona muito melhor - virtualmente, não há nenhuma perda de desempenho no throughput das consultas até cerca de 1000 clientes, e o fator da escala do MySQL (computado com a razão do throughput máximo para o thoughput de cada cliente.) foi de 180%. Temos observado resultados similares em sistemas com 4 processadores - virtualmente não há perda de desempenho quando o número de clientes é incrementado até 1000 e o fator da escala foi de 300%. Portanto para um servidor SMP muito carregado nós definitivamente recomendamos o kernel 2.4. Nós descobrimos que é essencial executar o processo `mysqld` com a mais alta prioridade possível no kernel 2.4 para obter performance máxima. Isto pode ser feito adicionando o comando `renice -20 $$` ao `mysqld_safe`. Nos nossos testes em uma máquina com 4 processadores, o aumento da prioridade nos deu 60% de aumento no throughput com 400 clientes.

Atualmente estamos tentando coletar mais informações sobre como o `MySQL` atua no kernel 2.4 em sistemas com 4 e 8 processadores. Se você tem acesso a um sistema deste porte e tem feito alguns benchmarks, por favor envie um email para `docs@mysql.com` com os resultados - iremos incluí-los neste manual.

Existe outro detalhe que afeta muito a performance do MySQL, especialmente em sistemas multi processados. A implementação de mutex em LinuxThreads na **glibc-2.1** é muito ruim para programas com várias threads que travam o mutex por um tempo curto. Em um sistema SMP, ironicamente, se você liga o MySQL com **LinuxThreads** sem modificações, removendo processadores da máquina, a performance do MySQL é melhorada em alguns casos. Para corrigir este comportamento, disponibilizamos um patch para **glibc 2.1.3**, em linuxthreads-2.1-patch (`http://www.mysql.com/Downloads/Linux/linuxthreads-2.1-patch`)

Com a **glibc-2.2.2**, o MySQL versão 3.23.36 irá usar o mutex adaptativo, que é muito melhor,mesmo que o patch na **glibc-2.1.3**. Avisamos, entretando, que sobre algumas condições, o código mutex no **glibc-2.2.2** overspins, que prejudica a performance do MySQL. A chance desta condição pode ser reduzida mudando a prioridade do processo `mysqld` para a prioridade mais alta. Nós também corrigimos o comportamento overspin com um patch, disponível em `http://www.mysql.com/Downloads/Linux/linuxthreads-2.2.2.patch`. Ele combina a correção do overspin, número máximo de threads e espaçamento das pilhas em um único patch. Você precisará aplicá-lo no diretório `linuxthreads` com `patch -p0 </tmp/linuxthreads-2.2.2.patch`. Esperamos que seja incluído de alguma forma nos futuros lançamentos da `glibc-2.2`. De qualquer forma, se você ligar com `glibc-2.2.2`, ainda será necessário corrigir `STACK_SIZE` e `PTHREAD_THREADS_MAX`. Temos esperanças que os padrões serão corrigidos para valores mais aceitáveis para configurações pesadasa do MySQL no futuro, então sua construção poderá ser reduzida a `./configure; make; make install`.

Recomendamos que você use os patches acima para construir uma versão estática especial de `libpthread.a` e use-a somente para ligações estáticas com o `MySQL`. Sabemos que os patches são seguros para o `MySQL` e pode melhorar significamente sua performance, mas não

podemos dizer nada sobre outras aplicações. Se você ligar outras aplicações coma a versão modificada da biblioteca ou construir uma versão alterada compartilhada e instalá-la no seu sistema, você estará fazendo por sua conta e risco e tenha atenção com outras aplicações que dependem de `LinuxThreads`.

Se você passar por problemas estranhos durante a instalação do MySQL ou com travamentos de alguns utilitários comuns, é muito provável que eles são relacionados a problemas de bibliotecas ou compilador. Se for este o caso, o uso de nosso binário será a solução.

Um problema conhecido com a distribuição binária é que com antigos sistemas Linux que usam `libc` (como o RedHat 4.x ou Slackware), você obterá alguns problemas não fatais com resolução de nomes. Veja Seção 2.6.2.1 [Binary notes-Linux], Página 141.

Quando estiver usando LinuxThreads você verá um mínimo de três processos em execução. Estes são de fato, threads. Existirá uma thread para o gerenciador LinuxThreads, uma thread para lidar com conexões e uma thread para tartar de alarmes e sinais.

Perceba que o kernel Linux e a biblioteca LinuxThread pode por padrão ter apenas 1024 threads. Isto significa que você pode ter até 1021 conexões ao MySQL em um sistema sem correção. A página `http://www.volano.com/linuxnotes.html` contém informações sobre como contornar este limite.

Se você ver um processo `mysqld` daemon finalizado com `ps`, isto normalmente significa que você encontrou um bug no MySQL ou que tenha uma tabela corrompida. Veja Seção A.4.1 [Crashing], Página 925.

Para obter um descarga do core no Linux se o `mysqld` finalizar com um sinal SIGSEGV, você pode iniciar o `mysqld` com a opção `--core-file`. Perceba que provavelmente você também precisará aumentar o `core file size` adicionando `ulimit -c 1000000` para `mysqld_safe` ou iniciar `mysqld_safe` com `--core-file-sizes=1000000`, Veja Seção 4.8.2 [safe_mysqld], Página 331.

Se você estiver ligando seu próprio cliente MySQL e obter o erro:

```
ld.so.1: ./my: fatal: libmysqlclient.so.4:
open failed: No such file or directory
```

Quando executá-los, o problema pode ser evitado com um dos seguintes métodos:

- Ligue o cliente com a seguinte opção (no lugar de `-Lpath`): `-Wl,r/path-libmysqlclient.so`.
- Copie `libmysqclient.so` para '`/usr/lib`'.
- Adicione o caminho do diretório onde `libmysqlclient.so` está localizado para a variável de ambiente `LD_RUN_PATH` antes de executar seu cliente.

Se você estiver usando o compilador Fujitsu `(fcc / FCC)` você terá alguns problemas compilando o MySQL porque os arquivos de cabeçalho Linux são muito orientados ao `gcc`.

A seguinte linha `configure` deve funcionar com `fcc/FCC`:

```
CC=fcc CFLAGS="-O -K fast -K lib -K omitfp -Kpreex -D_GNU_SOURCE \
-DCONST=const -DNO_STRTOLL_PROTO" CXX=FCC CXXFLAGS="-O -K fast -K lib \
-K omitfp -K preex --no_exceptions --no_rtti -D_GNU_SOURCE -DCONST=const \
-Dalloca=__builtin_alloca -DNO_STRTOLL_PROTO \
'-D_EXTERN_INLINE=static __inline'" ./configure --prefix=/usr/local/mysql \
--enable-assembler --with-mysqld-ldflags=-all-static --disable-shared \
--with-low-memory
```

2.6.2.1 Notas Linux para distribuições binárias

O MySQL necessita pelo menos do Linux versão 2.0

Aviso: Percebemos que alguns usuários do MySQL tiveram serios problemas de estabilidade com o MySQL e o kernel 2.2.14 do Linux. Se você estiver usando este kernel você deve atualizá-lo para o 2.2.19 (ou posterior) ou para o kernel 2.4. Se você tiver um gabinete multi-cpu, então você deve considerar seriamente o uso do kernel 2.4 uma vez que ele lhe trará uma melhora significante na velocidade.

A versão binária é ligada com `-static`, que significa que você normalmente não precisa se preocupar com qual versão das bibliotecas do sistema você tem. Você não precisa instalar LinuxThreads. Um programa ligado com a opção `-static` é um pouco maior que um programa ligado dinamicamente e também um pouco mais rápido (3-5%). Um problema, entretanto, é que você não pode usar funções definidas pelo usuário (UDF) com um programa ligado estaticamente. Se você for escrever ou usar funções UDF (isto é algo para programadores C ou C++), você deve compilar o MySQL, usando ligações dinamicas.

Se você estiver usando um sistema baseado em `libc` (em vez de um sistema `glibc2`), você, provavelmente, terá alguns problemas com resolução de nomes de máquinas e `getpwnam()` com a versão binária. (Isto é porque o `glibc` infelizmente depende de algumas bibliotecas externas para resolver nomes de máquinas e `getpwent()`, mesmo quando compilado com `-static`). Neste caso, você provavelmente obterá a seguinte mensagem de erro quando executar `mysql_install_db`:

```
Sorry, the host 'xxxx' could not be looked up
```

ou o seguinte erro quando você tentar executar `mysqld` com a opção `--user`:

```
getpwnam: No such file or directory
```

Você pode resolver este problema usando de um dos modos seguintes:

- Obtenha uma distribuição fonte do MySQL (uma distribuição RPM ou `tar.gz`) e a instale.
- Execute `mysql_install_db --force`; Isto não executará o teste `resolveip` no `mysql_install_db`. O lado ruim é que você não poderá usar nomes de máquinas nas tabelas de permissões; você deve usar números IP no lugar (exceto para `localhost`). Se você estiver usando uma release antiga do MySQL que não suporte `--force`, você deve remover o teste `resolveip` no `mysql_install` com um editor.
- Inicie `mysqld` com `su` no lugar de usar `--user`.

As distribuições binárias Linux-Intel e RPM do MySQL são configuradas para o máximo de desempenho possível. Nós sempre tentamos usar o compilador mais rápido e estável disponível.

Suporte MySQL ao Perl exige Perl Versão 5.004_03 ou mais novo.

Em algumas versões 2.2 do kernel Linux,você pode obter o erro `Resource temporarily unavailable` quando você faz várias novas conexões para um servidor `mysqld` sobre TCP/IP.

O problema é que o Linux tem um atraso entre o momento em que você fecha um socket TCP/IP até que ele seja realmente liberado pelo sistema. Como só existe espaço para um número finito de slots TCP/IP, você irá obter o erro acima se você tentar fazer muitas novas

conexões TCP/IP durante um pequeno tempo, como quando você executa o benchmark do MySQL '`test-connect`' sobre TCP/IP.

Nós enviamos emails sobre este problema várias vezes para diferentes listas de discussão Linux mas nunca conseguimos resolver este problema apropriadamente.

A única 'correção' conhecida , para este problema é usar conexões persistentes nos seus clientes ou usar sockets, se você estiver executando o servidor de banco de dados e clientes na mesma máquina. Nós experamos que o kernel `Linux 2.4` corrija este problema no futuro.

2.6.2.2 Notas Linux x86

O MySQL exige a versão 5.4.12 ou mais nova da `libc`. Sabe-se que funciona com a `libc` 5.4.46. A versão 2.0.6 e posterior da `glibc` também deve funcionar. Existem alguns problemas com os RPMs `glibc` da RedHat, portanto se você tiver problemas, confira se existe alguma atualização! Sabemos que os RPMs `glibc` 2.0.7-19 e 2.0.7-29 funcionam.

Se você estiver usando o Red Hat 8.0 ou uma nova biblioteca `glibc` 2.2.x, você deve iniciar o `mysqld` com a opção `--thread-stack=192K` (Use `-O thread_stack=192K antes do MySQL 4`). Se você não fizer isto o `mysqld` finlizará em `gethostbyaddr()` porque a nova biblioteca glibc exige mais de 128K de memória na pilha para esta chamada. Este tamanho de pilha é o padrão agora no MySQL 4.0.10 e acima.

Se você está usando o `gcc` 3.0 e acima para compilar o MySQL, você deve instalar a biblioteca `libstdc++v3` antes de compilar o MySQL; se você não fizer isto, você obterá um erro sobre um símbolo `__cxa_pure_virtual` perdido durante a ligação.

Em algumas distribuições Linux mais antigas, `configure` pode produzir um erro como este:

```
Syntax error in sched.h. Change _P to __P in the /usr/include/sched.h file.
See the Installation chapter in the Reference Manual.
```

Faça apenas o que a mensagem de erro diz e adicione um caractere sublinhado para a macro `_P` que tem somente um caractere sublinhado e então tente novamente.

Você pode obter alguns aviso quando estiver compilando; os mostrados abaixo podem ser ignorados:

```
mysqld.cc -o objs-thread/mysqld.o
mysqld.cc: In function 'void init_signals()':
mysqld.cc:315: warning: assignment of negative value '-1' to
'long unsigned int'
mysqld.cc: In function 'void * signal_hand(void *)':
mysqld.cc:346: warning: assignment of negative value '-1' to
'long unsigned int'
```

O `mysql.server` pode ser encontrado no diretório '`share/mysql`' sob o diretório de instalação MySQL ou no diretório '`support-files`' da árvore fonte MySQL.

Se o `mysqld` sempre descarregar um core na inicialização, o problema pode ser que você tenha um antigo '`/lib/libc.a`'. Tente renomeá-lo depois remova '`sql/mysqld`' e faça um novo `make install` e tente novamente. Este problema foi relatado em algumas instalações Slackware.

Se você obter o seguinte erro quando ligar o `mysqld`, significa que seu '`libg++.a`' não está instalado corretamente:

```
/usr/lib/libc.a(putc.o): In function '_IO_putc':
putc.o(.text+0x0): multiple definition of '_IO_putc'
```

Você pode evitar o uso de 'libg++.a' executando configure desta forma:

```
shell> CXX=gcc ./configure
```

2.6.2.3 Notas Linux SPARC

Em algumas implementações, readdir_r() está quebrada. O sintoma é que SHOW DATABASES sempre retorna um conjunto vazio. Isto pode ser corrigido removendo HAVE_READDIR_R do 'config.h' depois de configurar e antes de compilar.

2.6.2.4 Notas Linux Alpha

O MySQL Versão 3.23.12 é a primeira versão do MySQL que é testada no Linux-Alpha. Se você planeja usar o MySQL no Linux-Alpha, você deve ter certeza que possui esta versão ou mais nova.

Temos testado o MySQL no Alpha com nossos pacotes de benchmarks e testes, e ele parece funcinar muito bem.

Quando nós compilamos o binários MySQL padrões, nós estávamos usando SuSE 6.4, kernel 2.2.13-SMP, Compilador C Compaq (V6.2-504) e compilador C++ Compaq (V6.3-005) em uma máquina Compaq DS20 com um processador Alpha EV6.

Você pode encontrar os compiladores acima em http://www.support.compaq.com/alpha-tools. Usando estes compiladores, em vez do gcc, obtemos 9-14 % de melhora na performance com MySQL.

Note que a linha de configuração otimiza o binário para a CPU atual; isto significa que você só pode utilizar nosso binário se você tiver um processador Alpha EV6. Nós também compilamos estaticamente para evitar problemas de bibliotecas.

A partir das próximas distribuições adicionamos o parâmetro -arch generic em nossas opções de compilação, o qual assegura que o binário execute em todos os processadores Alpha. Nós também compilamos estaticamente para evitar problemas de bibliotecas.

```
CC=ccc CFLAGS="-fast -arch generic" CXX=cxx \
CXXFLAGS="-fast -arch generic -noexceptions -nortti" \
./configure --prefix=/usr/local/mysql --disable-shared \
--with-extra-charsets=complex --enable-thread-safe-client \
--with-mysqld-ldflags=-non_shared --with-client-ldflags=-non_shared
```

Se você deseja usar egcs a seguinte linha de configuração funcionou para nós:

```
CFLAGS="-O3 -fomit-frame-pointer" CXX=gcc \
CXXFLAGS="-O3 -fomit-frame-pointer -felide-constructors \
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
--disable-shared
```

Alguns problemas conhecidos quando executamos o MySQL no Linux-Alpha:

- Debugar aplicações baseadas em threads como o MysQL não irá funcionar com gdb 4.18. Você deve fazer download e usar o gdb 5.0!

- Se você tentar ligar o `mysqld` estaticamente quando usar o `gcc`, a imagem resultante irá descarregar um arquivo core no início. Em outras palavras, **NÃO** use `--with-mysqld-ldflags=-all-static` com `gcc`.

2.6.2.5 Notas Linux PowerPC

O MySQL deve funcionar no MkLinux com o mais novo pacote `glibc` (testado com `glibc` 2.0.7).

2.6.2.6 Notas Linux MIPS

Para ter o MySQL funcionando no Qube2. (Linux Mips), você precisará das bibliotecas `glibc` mais novas (Sabemos que `glibc-2.0.7.29C2` funciona). Você também deve usar o compilador `egcs` C++ (`egcs-1.0.2-9`, `gcc 2.95.2` ou mais nova).

2.6.2.7 Notas Linux IA-64

Para conseguir compilar o MySQL no Linux Ia64, usamos a seguinte linha de compilação: Usando `gcc-2.96`:

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc \
CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors \
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
"--with-comment=Official MySQL binary" --with-extra-charsets=complex
```

No Ia64 os binários do cliente MySQL estão usando bibliotecas compartilhadas. Isto significa se você instalar nossa distribuição binárias em algum outro lugar diferente de '`/usr/local/mysql`' você precisa modificar o '`/etc/ld.so.conf`' ou adicionar o caminho da o diretório onde está localizado o '`libmysqlclient.so`' na variável de ambiente `LD_LIBRARY_PATH`.

Veja Seção A.3.1 [Erros de ligação], Página 922.

2.6.3 Notas Solaris

No Solaris, você deve ter problemas mesmo antes de descompactar a distribuição MySQL! O `tar` do Solaris não pode tratar grandes nomes de arquivos, portanto você pode ver um erro deste tipo quando descompactar o MySQL:

```
x mysql-3.22.12-beta/bench/Results/ATIS-mysql_odbc-NT_4.0-cmp-db2,informix,ms-sql,m
tar: directory checksum error
```

Neste caso, você deve usar o GNU `tar` (`gtar`) para desempacotar a distribuição. Você pode encontrar uma cópia pré-compilada para Solaris em `http://www.mysql.com/downloads/os-solaris.html`.

As threads nativas da Sun funcionam somente no Solaris 2.5 e superior. Para a versão 2.4 e anteriores, o MySQL irá automaticamente usar MIT-pthreads. Veja Seção 2.3.6 [MIT-pthreads], Página 106.

Se você obter o seguinte erro de configure:

```
checking for restartable system calls... configure: error can not run test
programs while cross compiling
```

Isto significa que alguma coisa está errada com a instalação de seu compilador! Neste caso você deve atualizar seu compilador para uma versão mais nova. Você também pode resolver este problema inserindo a seguinte linha no arquivo '`config.cache`':

```
ac_cv_sys_restartable_syscalls=${ac_cv_sys_restartable_syscalls='no'}
```

Se você está usando Solaris em um SPARC, o compilador recomendado é o `gcc` 2.95.2. Você pode encontrá-lo em `http://gcc.gnu.org/`. Perceba que `egcs` 1.1.1 e `gcc` 2.8.1 não são estáveis no SPARC!

A linha do `configure` recomendado quando usando `gcc` 2.95.2 é:

```
CC=gcc CFLAGS="-O3" \
CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory --enable-assembler
```

Se você possui um ultra sparc, você pode obter 4% a mais de performance adicionando "-mcpu=v8 -Wa,-xarch=v8plusa" para a CFLAGS e CXXFLAGS.

Se você possui o compilador Sun Workshop (Fortre) 5.3 (ou mais novo), você pode executar `configure` da seguinte forma:

```
CC=cc CFLAGS="-Xa -fast -native -xstrconst -mt" \
CXX=CC CXXFLAGS="-noex -mt" \
./configure --prefix=/usr/local/mysql --enable-assembler
```

Você pode criar um binário de 64 bits usando o compilador Forte da Sun com os seguintes parâmetros de compilação:

```
CC=cc CFLAGS="-Xa -fast -native -xstrconst -mt -xarch=v9" \
CXX=CC CXXFLAGS="-noex -mt -xarch=v9" ASFLAGS="-xarch=v9" \
./configure --prefix=/usr/local/mysql --enable-assembler
```

Para criar um binário de 64 bits do Solaris usando `gcc`, e `-m64` para `CFLAGS` e `CXXFLAGS`. Note que isto só funciona com o MySQL 4.0 e acima - o MySQL 3.23 não inclui as modificações exigidas para suportar isto.

No benchmark do MySQL, conseguimos um aumento de velocidade de 4% em um UltraSPARC usando o Forte 5.0 no modo 32 bit em comparação com o uso do gcc 3.2 com o parametro -mcpu.

Se você criar um binário de 64 bits, ele será 4$ mais lento que o binário de 32 bits, mas o `mysqld` poderá tratar mais threads e memória.

Se você tiver um problema com `fdatasync` ou `sched_yield`, você pode corrigir isto adicionando `LIBS=-lrt` para a linha de configuração

O seguinte paragráfo é relevante somente para compiladores mais antigos que o WorkShop 5.3:

Você também pode ter que editar o script `configure` para alterar esta linha:

```
#if !defined(__STDC__) || __STDC__ != 1
```

para isto:

```
#if !defined(__STDC__)
```

Se você ligar `__STDC__` com a opção `-Xc`, o compilador Sun não pode compilar com o arquivo de cabeçalho '`pthread.h`' do Solaris. Isto é um bug da Sun (compilador corrompido ou arquivo include corrompido).

Se o `mysqld` emitir a mensagem de erro mostrada abaixo quando você executá-lo, você deve tentar compilar o MySQL com o compilador Sun sem habilitar a opção multi-thread (`-mt`):

```
libc internal error: _rmutex_unlock: rmutex not held
```

Adicione `-mt` a `CFLAGS` e `CXXFLAGS` e tente novamente.

Se você estiver usando a versão SFW do gcc (que vem com o Solaris 8), você deve adicionar '`/opt/sfw/lib`' a variável de ambiente `LD_LIBRARY_PATH` antes de executar a configuração.

Se você estiver usando o gcc disponível em `sunfreeware.com`, você pode ter muitos problemas. Você deve recompilar o gcc e GNU binutils na máquina que você o executará para evitar qualquer problema.

Se você obter o seguinte erro quando estiver compilando o MySQL com `gcc`, significa que seu `gcc` não está configurado para sua versão de Solaris:

```
shell> gcc -O3 -g -O2 -DDBUG_OFF  -o thr_alarm ...
./thr_alarm.c: In function ‘signal_hand’:
./thr_alarm.c:556: too many arguments to function ‘sigwait’
```

A coisa apropriada para fazer neste caso é obter a versão mais nova do `gcc` e compilá-lo com seu compilador `gcc` atual! Ao menos para o Solaris 2.5, a maioria das versões binárias de `gcc` tem arquivos inúteis e antigos que irão quebrar todos programas que usam threads (e possivelmente outros programas)!

O Solaris não fornece versões estáticas de todas bibliotecas de sistema (`libpthreads`) e `libdl`), portanto você não pode compilar o MySQL com `--static`. Se você tentar fazer isto, receberá o erro:

```
ld: fatal: library -ldl: not found
```

ou

```
undefined reference to ‘dlopen’
```

ou

```
cannot find -lrt
```

Se vários processos tentar conectar muito rapidamente ao `mysqld`, você verá este erro no log do MySQL:

```
Error in accept: Protocol error
```

Você deve tentar iniciar o servidor com a opção `--set-variable back_log=50` como uma solução para esta situação. Note que `--set-variable=nome=valor` e `-O nome=valor` está obsoleto desde o MySQL 4.0. Use apenas `--back_log=50`. Veja Seção 4.1.1 [Opções da linha de comando], Página 207.

Se você está ligando seu próprio cliente MySQL, você deve obter o seguinte erro quando tentar executá-lo:

```
ld.so.1: ./my: fatal: libmysqlclient.so.#:
```

```
open failed: No such file or directory
```

O problema pode ser evitado por um dos seguintes métodos:

- Ligue o cliente com a seguinte opção (em vez de `-Lpath`): `-Wl,r/full-path-to-libmysqlclient.so`.
- Copie o arquivo '`libmysqclient.so`' para '`/usr/lib`'.
- Adicione o caminho do diretório onde '`libmysqlclient.so`' está localizado à variável de ambiente `LD_RUN_PATH` antes de executar seu cliente.

Se você tiver problemas com o configure tentando ligar com `-lz` e você não tem a `zlib` instalada, você terá duas opções:

- Se você deseja usar o protocol de comunição de compactado você precisará obter e instalar a zlib from ftp.gnu.org.
- Configure com `--with-named-z-libs=no`.

Se você estiver usando o gcc e tiver problemas carregando funções `UDF` no MySQL, tente adicionar `-lgcc` para a linha de ligação para a função `UDF`.

Se você deseja que o MySQL inicie automaticamente, você pode copiar '`support-files/mysql.server`' para '`/etc/init.d`' e criar um link simbólico para ele, chamado '`/etc/rc.3.d/S99mysql.server`'.

Como o Solaris não suporta core files para aplicações `setuid()`, você não pode obter um core file do `mysqld` se você estiver usando a opção `--user`.

2.6.3.1 Notas Solaris 2.7/2.8

Você pode utilizar normalmente um binário Solaris 2.6 no Solaris 2.7 e 2.8. A maioria dos detalhes do Solaris 2.6 também se aplicam ao Solaris 2.7 e 2.8.

Note que o MySQL versão 3.23.4 e superiores devem estar aptos para autodetectar novas versões do Solaris e habilitar soluções para os problemas seguintes!

Solaris 2.7 / 2.8 tem alguns bugs nos arquivos include. Você pode ver o seguinte erro quando você usa o `gcc`:

```
/usr/include/widec.h:42: warning: 'getwc' redefined
/usr/include/wchar.h:326: warning: this is the location of the previous
definition
```

Se isto ocorrer, você pode fazer o seguinte para corrigir o problema:

Copie `/usr/include/widec.h` para `.../lib/gcc-lib/os/gcc-version/include` e mude a linha 41 :

```
#if     !defined(lint) && !defined(__lint)

para

#if     !defined(lint) && !defined(__lint) && !defined(getwc)
```

Uma alternativa é editar o '`/usr/include/widec.h`' diretamente. Desta forma, depois de fazer a correção, você deve remover o '`config.cache`' e executar o `configure` novamente !

Se você obter erros como estes quando você executar o `make`, é porque o `configure` não encontrou o arquivo '`curses.h`' (provavelmente devido ao erro no arquivo '`/usr/include/widec.h`'):

```
In file included from mysql.cc:50:
/usr/include/term.h:1060: syntax error before `,'
/usr/include/term.h:1081: syntax error before `;'
```

A solução para isto é fazer uma das seguintes opções:

- Configure com `CFLAGS=-DHAVE_CURSES_H CXXFLAGS=-DHAVE_CURSES_H ./configure`.
- Edite o '`/usr/include/widec.h`' como indicado acima e re-execute o configure.
- Remova a linha `#define HAVE_TERM` do arquivo '`config.h`' e execute `make` novamente.

Se o seu ligador tiver problemas para encontrar o `-lz` quando ligar ao seu programa cliente, provavelmente o problema é que seu arquivo '`libz.so`' está instalado em '`/usr/local/lib`'. Você pode corrigir isto usando um dos seguintes métodos:

- Adicione '`/usr/local/lib`' ao `LD_LIBRARY_PATH`.
- Adicione um link para '`libz.so`' a partir de '`/lib`'.
- Se você estiver usando o Solaris 8, você pode instalar a zlib opcional do CD de distribuição do Solaris 8.
- Configure o MySQL com a opção `--with-named-z-libs=no`.

2.6.3.2 Notas Solaris x86

No Solaris 8 no x86, `mysqld` irá descarregar um core se você executar um 'strip' no mesmo.

Se você estiver usando `gcc` ou `egcs` no Solaris X86 e você tiver problemas com descarregos de core, você deve utilizar o seguinte comando `configure`:

```
CC=gcc CFLAGS="-O3 -fomit-frame-pointer -DHAVE_CURSES_H" \
CXX=gcc \
CXXFLAGS="-O3 -fomit-frame-pointer -felide-constructors -fno-exceptions -fno-rtti -
./configure --prefix=/usr/local/mysql
```

Isto irá evitar problemas com a biblioteca `libstdc++` e com exceções C++.

Se isto não ajudar, você pode compilar uma versão com debug e executá-lo com um arquivo de ratreamento (trace) ou sob `gdb`. Veja Seção E.1.3 [Using gdb on mysqld], Página 1081.

2.6.4 Notas BSD

Esta seção fornece informação para os vários tipos de BSD, assim como a versão específica para eles.

2.6.4.1 Notas FreeBSD

FreeBSD 4.x ou mais novo é recomendado para executação do MySQL uma vez que o pacote thread é muito mais integrado.

A mais fácil e portanto a forma preferida para instalá-lo é usar as portas mysql-server e mysql-client disponíveis em `http://www.freebsd.org`.

Usando-as você obtem:

- Um MySQL funcional, com todas as otimizações conhecidas para trabalhar na sua versão habilitada do FreeBSD.
- Configuração e construção automática.
- Scripts de inicialização instalados em /usr/local/etc/rc.d.
- Habilidade para ver quais arquivos estão instalados com pkg_info -L. E para remover todos com pkg_delete se você não quiser mais o MySQL na máquina.

É recomendado que você utilize MIT-pthreads no FreeBSD 2.x e threads nativas nas Versões 3 e superiores. É possível executar com threads nativas em algumas versões antigas (2.2.x) mas você pode encontrar problemas ao finalizar o `mysqld`.

Infelizmente algumas chamadas de funções no FreeBSD ainda não são totalmente seguras com threads, principalmente a função `gethostbyname()`, que é usada pelo MySQL para converter nomes de máquinas em endereços IPs. Sob certas circunstâncias, o processo `mysqld` irá criar repentinamente um carga de CPU de 100% e ficará sem resposta. Se você se deparar com isto, tente iniciar o MySQL usando a opção `--skip-name-resolve`.

Alternativamente, você pode ligar o MySQL no FreeBSD 4.x com a biblioteca LinuxThreads, que evita uns poucos problemas que a implementação da thread nativa do FreeBSD tem. Para uma comparação muito boa do LinuxThreads vs. threads nativas dê uma olhada no artigo "FreeBSD or Linux for your MySQL Server?" de Jeremy Zawodny em `http://jeremy.zawodny.com/blog/archives/000697.html`

Os problemas conhecidos usando LinuxThreads na FreeBSD são:

- `wait_timeout` não está funcionando (provavemente problema de manipulação do signal em FreeBSD/LinuxThreads). Isto deveria ter sido corrigido no FreeBSD 5.0. O sintome á que conexões persistentes podem se manter por **um longo** tempo sem serem fechadas.

O '`Makefile`' do MySQL necessita o GNU make (`gmake`) para funcionar. Se você deseja compilar o MySQL, antes você precisará instalar o GNU `make`.

Tenha certeza que sua configuração de resolução de nomes esteja correta. De outra forma você vai ter atrasos na resolução ou falhas quando conectar ao `mysqld`.

Tenha certeza que a entrada `localhost` no arquivo '`/etc/hosts`' esteja correta (de outra forma você irá ter problemas conectando ao banco de dados). O arquivo '`/etc/hosts`' deve iniciar com a linha:

```
127.0.0.1       localhost localhost.seu.dominio
```

O modo recomendado de compilar e instalar o MySQL no FreeBSD com gcc (2.95.2 e acima) é:

```
CC=gcc CFLAGS="-O2 -fno-strength-reduce" \
CXX=gcc CXXFLAGS="-O2 -fno-rtti -fno-exceptions -felide-constructors \
-fno-strength-reduce" \
./configure --prefix=/usr/local/mysql --enable-assembler
gmake
gmake install
./scripts/mysql_install_db
cd /usr/local/mysql
./bin/mysqld_safe &
```

Se você percber que o `configure` usará MIT-pthreads, você de ler as notas sobre MIT-pthreads. Veja Seção 2.3.6 [MIT-pthreads], Página 106.

Se o `make install` não puder encontrar '`/usr/include/pthreads`', é porque o `configure` não detectou que você precisava de MIT-pthreads. Isto é corrigido executando estes comandos:

```
shell> rm config.cache
shell> ./configure --with-mit-threads
```

O FreeBSD é também conhecido por ter um limite muito baixo para o manipulador de arquivos. Veja Seção A.2.17 [Not enough file handles], Página 921. Descomente a seção ulimit -n no mysqld_safe ou aumente os limites para o usuário `mysqld` no /etc/login.conf (e reconstrua-o com cap_mkdb /etc/login.conf). Também tenha certeza que você configurou a classe apropriada para este usuário no arquivo de senhas (password) se você não estiver usando o padrão (use: chpass nome_usuario_mysqld). Veja Seção 4.8.2 [`mysqld_safe`], Página 331.

Se você tiver muita memória você deve considerar em reconstruir o Kernel para permitir o MySQL de usar mais de 512M de RAM. Dê uma olhada na `opção MAXDSIZ` na arquivo de configuração LINT para maiores informações.

Se você tiver problemas com a data atual no MySQL, configurar a variável `TZ` provavelmente ajudará. Veja Apêndice F [Environment variables], Página 1092.

Para obter um sistema seguro e estável você deve usar somente kernels FreeBSD que estejam marcados com `-STABLE`.

2.6.4.2 Notas NetBSD

Para compilar no NetBSD você precisa do GNU `make`. De outra forma o compilador quebraria quando o `make` tentasse executar `lint` em arquivos C++.

2.6.4.3 Notas OpenBSD

No OpenBSD Versão 2.5, você pode compilar o MySQL com threads nativas com as seguintes opções:

```
CFLAGS=-pthread CXXFLAGS=-pthread ./configure --with-mit-threads=no
```

2.6.4.4 Notas OpenBSD 2.8

Nossos usuários relataram que o OpenBSD 2.8 tem um bug nas threads que causa problemas com o MySQL. Os desenvolvedores do OpenBSD já corrigiram o problema, mas em 25 de Janeiro de 2001 a correção foi disponível apenas no ramo "-current". Os sintomas deste bug nas threads são: resposta lenta, alta carga, alto uso de CPU e quedas do servidor.

Se você obter um erro como `Error in accept:: Bad file descriptor` ou erro 9 ao tentar abrir tabelas ou diretórios, o problema é provavelmente que você não alocou memória suficiente para os descritores de arquivo do MySQL.

Neste caso tente iniciar o `mysqld_safe` como `root` com as seguintes opções:

```
shell> mysqld_safe --user=mysql --open-files-limit=2048 &
```

2.6.4.5 Notas BSDI Versão 2.x

Se você obter o seguinte erro quando estiver compilando o MySQL, seu valor `ulimit` para memória virtual é muito baixo:

```
item_func.h: In method `Item_func_ge::Item_func_ge(const Item_func_ge &)':
item_func.h:28: virtual memory exhausted
make[2]: *** [item_func.o] Error 1
```

Tente usar `ulimit -v 80000` e executar o `make` novamente. Se isto não funcionar e você estiver usando o `bash`, tente trocar para `csh` ou `sh`; alguns usuários BSDI relataram problemas com `bash` e `ulimit`.

Se você utiliza `gcc`, você pode também ter de usar a opção `--with-low-memory` para o `configure` estar apto a compilar o '`sql_yacc.cc`'.

Se você tiver problemas com a data atual no MySQL, configurar a variável `TZ` provavelmente ajudará. Veja Apêndice F [Environment variables], Página 1092.

2.6.4.6 Notas BSD/OS Versão 3.x

Atualize para BSD/OS Versão 3.1. Se isto não for possível, instale BSDIpatch M300-038.

Use o seguinte comando quando configurar o MySQL:

```
shell> env CXX=shlicc++ CC=shlicc2 \
       ./configure \
           --prefix=/usr/local/mysql \
           --localstatedir=/var/mysql \
           --without-perl \
           --with-unix-socket-path=/var/mysql/mysql.sock
```

O comeando seguinte também funciona:

```
shell> env CC=gcc CXX=gcc CXXFLAGS=-O3 \
       ./configure \
           --prefix=/usr/local/mysql \
           --with-unix-socket-path=/var/mysql/mysql.sock
```

Você pode alterar as localizações dos diretórios se você desejar, ou apenas usar os padrões não especificando nenhuma localização.

Se você tiver problemas com performance sob alta carga, tente usar a opção `--skip-thread-priority` para `mysqld`! Isto irá executar todas as threads com a mesma prioridade; no BSDI versão 3.1, isto fornece melhor performance (pelo menos até o BSDI corrigir seu organizador de threads).

Se você obter o erro `virtual memory exhausted` enquanto estiver compilando, deve tentar usar `ulimit -v 80000` e executar `make` novamente. Se isto não funcionar e você estiver usando `bash`, tente trocar para `csh` ou `sh`; alguns usuários BSDI relataram problemas com `bash` e `ulimit`.

2.6.4.7 Notas BSD/OS Versão 4.x

O BSDI Versão 4.x tem alguns bugs relacionados às threads. Se você deseja usar o MySQL nesta versão, você deve instalar todas as correções relacionadas às threads. Pelo menos a M400-23 deve estar instalada.

Em alguns sistemas BSDI versão 4.x, você pode ter problemas com bibliotecas compartilhadas. O sintoma é que você não pode executar nenhum programa cliente, por exemplo, `mysqladmin`. Neste caso você precisa reconfigurar o MySQL, para ele não usar bibliotecas compartilhadas, com a opção `--disable-shared`.

Alguns clientes tiveram problemas no BSDI 4.0.1 que o binário do `mysqld` não conseguia abrir tabelas depois de um tempo em funcionamento. Isto é porque alguns bugs relacionados a biblioteca/sistema fazem com que o `mysqld` altere o diretório atual sem nenhuma informação!

A correção é atualizar para a 3.23.34 ou depois de executar `configure` remova a linha `$define HAVE_REALPATH` de `config.h` antes de executar o make.

Perceba que com isso você não pode fazer um link simbólico de um diretório de banco de dados para outro diretório ou fazer um link simbólico a uma tabela para outro banco de dados no BSDI! (Criar um link simbólico para outro disco funciona).

2.6.5 Notas Mac OS X

2.6.5.1 Mac OS X 10.x

O MySQL deve funcionar sem problemas no Mac OS X 10.x (Darwin). Você não precisa dos patches pthread para este SO.

Isto também se aplica ao Mac OS X 10.x Server. A compilação para a plataforma Server é a mesma para a versão cliente do Mac OS X. No entanto note que o MySQL vem preinstalado no Servidor !

Nosso binário para Mac OS X é compilado no Darwin 6.3 com a seguinte linha de configuração:

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc \
CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors \
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
--with-extra-charsets=complex --enable-thread-safe-client \
--enable-local-infile --disable-shared
```

Veja Seção 2.1.3 [Instalação do Mac OS X], Página 71.

2.6.5.2 Mac OS X Server 1.2 (Rhapsody)

Antes de tentar configurar o MySQL no MAC OS X server 1.2 (aka Rhapsody), primeiro você deve instalar o pacote pthread encontrado em: `http://www.prnet.de/RegEx/mysql.html`.

Veja Seção 2.1.3 [Instalação do Mac OS X], Página 71.

2.6.6 Notas de Outros Unix

2.6.6.1 Notas HP-UX para distribuições binárias

Alguma das distribuições binárias do MySQL para HP-UX é distribuida como um arquivo depot da HP e como um arquivo tar. Para usar o arquivo depot você deve estar executando pelo menos o HP-UX 10.x para ter acesso às ferramentas de arquivos depot da HP.

A versão HP do MySQL foi compilada em um servidor HP 9000/8xx sob HP-UX 10.20, usando MIT-pthreads. Sob esta configuração o MySQL funciona bem. O MySQL Versão 3.22.26 e mais novas também podem ser construidas com o pacote thread nativo da HP.

Outras configurações que podem funcionar:

- HP 9000/7xx executando HP-UX 10.20+
- HP 9000/8xx executando HP-UX 10.30

As seguintes configurações definitivamente não funcionarão:

- HP 9000/7xx ou 8xx executando HP-UX 10.x where x < 2
- HP 9000/7xx ou 8xx executando HP-UX 9.x

Para instalar a distribuição, utilze um dos comandos abaixo, onde `/path/to/depot` é o caminho completo do arquivo depot:

- Para instalar tudo, incluindo o servidor, cliente e ferramentas de desenvolvimento:

```
shell> /usr/sbin/swinstall -s /path/to/depot mysql.full
```

- Para instalar somente o servidor:

```
shell> /usr/sbin/swinstall -s /path/to/depot mysql.server
```

- Para instalar somente o pacote cliente:

```
shell> /usr/sbin/swinstall -s /path/to/depot mysql.client
```

- Para instalar somente as ferramentas de desenvolvimento:

```
shell> /usr/sbin/swinstall -s /path/to/depot mysql.developer
```

O depot copia os binários e bibliotecas em '`/opt/mysql`' e dados em '`/var/opt/mysql`'. O depot também cria as entradas apropriadas em '`/etc/init.d`' e '`/etc/rc2.d`' para iniciar o servidor automaticamente na hora do boot. Obviamente, para instalar o usuário deve ser o `root`.

Para instalar a distribuição HP-UX tar.gz, você deve ter uma cópia do GNU `tar`.

2.6.6.2 Notas HP-UX Versão 10.20

Existem alguns pequenos problemas quando compilamos o MySQL no HP-UX. Nós recomendamos que você use o `gcc` no lugar do compilador nativo do HP-UX, porque o `gcc` produz um código melhor!

Nós recomendamos o uso do `gcc` 2.95 no HP-UX. Não utilize opções de alta otimização (como -O6) ja que isto pode não ser seguro no HP-UX.

A seguine linha do `configure` deve funcionar com o gcc 2.95:

```
CFLAGS="-I/opt/dce/include -fpic" \
CXXFLAGS="-I/opt/dce/include -felide-constructors -fno-exceptions \
-fno-rtti" CXX=gcc ./configure --with-pthread \
--with-named-thread-libs='-ldce' --prefix=/usr/local/mysql --disable-shared
```

A seguinte linha do `configure` deve funcionar com o gcc 3.1:

```
CFLAGS="-DHPUX -I/opt/dce/include -O3 -fPIC" CXX=gcc \
CXXFLAGS="-DHPUX -I/opt/dce/include -felide-constructors -fno-exceptions \
-fno-rtti -O3 -fPIC" ./configure --prefix=/usr/local/mysql \
--with-extra-charsets=complex --enable-thread-safe-client \
--enable-local-infile  --with-pthread \
--with-named-thread-libs=-ldce --with-lib-ccflags=-fPIC
--disable-shared
```

2.6.6.3 Notas HP-UX Versão 11.x

Para HP-UX Versão 11.x nós recomendamos o MySQL Versão 3.23.15 ou posterior.

Por causa de alguns bugs críticos nas bibliotecas padrão do HP-UX, você deve instalar as seguintes correções antes de tentar executar o MySQL no HP-UX 11.0:

```
PHKL_22840 Streams cumulative
PHNE_22397 ARPA cumulative
```

Isto irá resolver um problema que tem como retorno `EWOLDBLOCK` de `recv()` e `EBADF` de `accept()` em aplicações threads.

Se você estiver usando `gcc` 2.95.1 em um sistema HP-UX 11.x sem correções, você obterá o erro:

```
In file included from /usr/include/unistd.h:11,
                 from ../include/global.h:125,
                 from mysql_priv.h:15,
                 from item.cc:19:
/usr/include/sys/unistd.h:184: declaration of C function ...
/usr/include/sys/pthread.h:440: previous declaration ...
In file included from item.h:306,
                 from mysql_priv.h:158,
                 from item.cc:19:
```

O problema é que o HP-UX não define consistentemente a `pthreads_atfork()`. Ela tem protótipos coflitantes em '`/usr/include/sys/unistd.h`':184 e '`/usr/include/sys/pthread.h`':440 (detalhes abaixo).

Uma solução é copiar '`/usr/include/sys/unistd.h`' em '`mysql/include`' e editar '`unistd.h`' alterando-o para coincidir com a definição em '`pthread.h`'. Aqui está o diff:

```
183,184c183,184
<       extern int pthread_atfork(void (*prepare)(), void (*parent)(),
<                                                 void (*child)());
---
>       extern int pthread_atfork(void (*prepare)(void), void (*parent)(void),
>                                                 void (*child)(void));
```

Depois disto, a seguinte linha configure deve funcionar:

```
CFLAGS="-fomit-frame-pointer -O3 -fpic" CXX=gcc \
CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti -O3" \
./configure --prefix=/usr/local/mysql --disable-shared
```

Segue algumas inforamações que um usuário do HP-UX Versão 11.x nos enviou sobre compilação do MySQL com o compilador HP-UX:

```
CC=cc CXX=aCC CFLAGS=+DD64 CXXFLAGS=+DD64 ./configure --with-extra-character-set=co
```

Você pode ignorar qualquer erro do tipo:

```
aCC: warning 901: unknown option: '-3': use +help for online documentation
```

Se você obter o seguinte erro do `configure`

```
checking for cc option to accept ANSI C... no
configure: error: MySQL requires a ANSI C compiler (and a C++ compiler).
Try gcc. See the Installation chapter in the Reference Manual.
```

Confira se você não tem o caminho para o compilador K&R antes do caminho para o compilador C e C++ do HP-UX.

Outra razão para não estar compilando é você não definir o parâmetro `+DD64` acima.

Outra possibilidade para o HP-UX 11 é usar o binário MySQL para HP-UX 10.20. Recebemos relatos de alguns usuários de que esses binários funcionam bem no HP-UX 11.00. Se você encontrar problemas, verifique o nível do pacth de seu HP-UX.

2.6.6.4 Notas IBM-AIX

Detecção automática de `xlC` está faltando no Autoconf, portando um comando `configure` deste tipo é necessário quando estiver compilando o MySQL (Este exemplo usa o compilador IBM):

```
export CC="xlc_r -ma -O3 -qstrict -qoptimize=3 -qmaxmem=8192 "
export CXX="xlC_r -ma -O3 -qstrict -qoptimize=3 -qmaxmem=8192"
export CFLAGS="-I /usr/local/include"
export LDFLAGS="-L /usr/local/lib"
export CPPFLAGS=$CFLAGS
export CXXFLAGS=$CFLAGS

./configure --prefix=/usr/local \
--localstatedir=/var/mysql \
--sysconfdir=/etc/mysql \
--sbindir='/usr/local/bin' \
--libexecdir='/usr/local/bin' \
--enable-thread-safe-client \
--enable-large-files
```

Acima estão as opções usadas para compilar a distribuição MySQL que pode ser encontrada em `http://www-frec.bull.com/`.

Se você alterar o `-O3` para `-O2` na linha de configuração acima, você também deve remover a opção `-qstrict` (isto é uma limitação no compilador C da IBM).

Se você estiver usando `gcc` ou `egcs` para compilar o MySQL, você **DEVE** usar a opção `-fno-exceptions`, já que o manipulador de exceções no `gcc`/`egcs` não é seguro para threads! (Isto

foi testado com `egcs` 1.1). Existem também alguns problemas conhecidos com o assembler da IBM que pode gerar código errado quando usado com gcc.

Nós recomendamos a seguinte linha do `configure` com `egcs` e `gcc 2.95` no AIX:

```
CC="gcc -pipe -mcpu=power -Wa,-many" \
CXX="gcc -pipe -mcpu=power -Wa,-many" \
CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory
```

O `-Wa,-many` é necessário para o compilador ser bem sucedido. IBM está ciente deste problema mas não está com pressa de corrigí-lo devido ao fato do problema poder ser contornado. Nós não sabemos se o `-fno-exceptions` é necessário com `gcc 2.9.5`, mas como o MySQL não utiliza exceções e a opção acima gera código mais rápido, recomendamos que você sempre use esta opção com o `egcs/gcc`.

Se você tiver algum problema com código assembler tente alterar o -mcpu=xxx para o seu processador. Normalmente power2, power ou powerpc podem ser usados, de uma maneira alternativa você pode precisar usar 604 ou 604e. Não tenho certeza mas acredito que usar "power" deve satisfazer a maioria dos casos, mesmo em uma máquina power2.

Se você não sabe qual é o seu processador, utilize o comando "uname -m", isto irá fornecer a você uma string que parece com "000514676700", com um formato de xxyyyyyymmss onde xx e ss são sempre 0s, yyyyyy é o ID único do sistema e mm é o ID da CPU Planar. Uma tabela destes valores podem ser encontrados em `http://publib.boulder.ibm.com/doc_link/en_US/a_doc_lib/cmds/aixcmds5/uname.htm`. Isto irá lhe fornecer um tipo de máquina e um modelo de máquina que você pode usar para determinar que tipo de cpu você tem.

Se você tiver problemas com sinais (MySQL finaliza sem notificação sob alta carga) você pode ter encontrado um bug de SO com threads e sinais. Neste caso você pode dizer ao MySQL para não usar sinais configurando-o com:

```
shell> CFLAGS=-DDONT_USE_THR_ALARM CXX=gcc \
       CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti \
       -DDONT_USE_THR_ALARM" \
       ./configure --prefix=/usr/local/mysql --with-debug --with-low-memory
```

Isto não afeta a performance do MySQL, mas tem o efeito colateral que você não pode matar clientes que estão "dormindo" em uma conexão com `mysqladmin kill` ou `mysqladmin shutdown`. Neste caso, o cliente morrerá quando ele chegar no próximo comando.

Em algumas versões do AIX, ligando com `libbind.a` faz o `getservbyname` descarregar core. Isto é erro no AIX e deve ser relatado para a IBM.

Para o AIX 4.2.1 e gcc você tem que fazer as seguintes alterações.

Depois de configurar, edite o '`config.h`' e '`include/my_config.h`' e altere a linha que diz

```
#define HAVE_SNPRINTF 1
```

para

```
#undef HAVE_SNPRINTF
```

E finalmente, no '`mysqld.cc`' você precisa adicionar um protótipo para initgroups.

```
#ifdef _AIX41
extern "C" int initgroups(const char *,int);
```

```
#endif
```

Se você precisar se alocar muita memória para o processo mysqld, não é suficiente apenas definir 'ulimit -d unlimited'. Você também deve configurar no `mysqld_safe` algo do tipo:

```
export LDR_CNTRL='MAXDATA=0x80000000'
```

Você pode encontrar mais sobre o uso de muita memória em: `http://publib16.boulder.ibm.com/pseries/en_US/aixprggd/genprogc/lrg_prg_support.htm`.

2.6.6.5 Notas SunOS 4

No SunOS 4, é necessário a MIT-pthreads para compilar o MySQL, o que significa que você precisa do GNU `make`.

Alguns sistemas SunOS 4 tem problemas com bibliotecas dinâmicas e `libtool`. Você pode usar a seguinte linha do `configure` para evitar este problema:

```
shell> ./configure --disable-shared --with-mysqld-ldflags=-all-static
```

Quando compilando `readline`, você pode obter alguns avisos sobre definições duplicadas que podem ser ignoradas.

Ao compilar o `mysqld`, vão existir alguns alertas sobre `implicit declaration of function` que também podem ser ignoradas.

2.6.6.6 Notas Alpha-DEC-UNIX (Tru64)

Se você está usando o egcs 1.1.2 no Digital Unix, você atualizar par o gcc 2.95.2, já que o egcs no DEC tem vários erros graves !

Quando compilando programas com threads no Digital Unix, a documentação recomenda usar a opção `-pthread` para `cc` e `cxx` e as bibliotecas `-lmach -lexc` (em adição para `-lpthread`). Você deve executar o `configure` parecido com isto:

```
CC="cc -pthread" CXX="cxx -pthread -O" \
./configure --with-named-thread-libs="-lpthread -lmach -lexc -lc"
```

Quando compilando o `mysqld`, você deve ver alguns avisos como estes:

```
mysqld.cc: In function void handle_connections()':
mysqld.cc:626: passing long unsigned int *' as argument 3 of
accept(int,sockadddr *, int *)'
```

Você pode ignorar estes altertas com segurança. Eles ocorrem porque o `configure` só pode detectar erros e não alertas.

Se você inicia o servidor diretamente da linha de comando, você pode ter problemas com a finalização do servidor ao sair (log out). (Quando você sai, seu processo superior recebe um sinal `SIGHUP`.) Se isto acontecer, tente iniciar o servidor desta forma:

```
shell> nohup mysqld [options] &
```

`nohup` faz com que o comando que o segue ignore qualquer sinal `SIGHUP` enviado pelo terminal. De forma alternativa, inicie o servidor executando `mysqld_safe`, o qual invoca o `mysqld` usando `nohup` por você. Veja Seção 4.8.2 [mysqld_safe], Página 331.

Se você tiver problemas quando compilar mysys/get_opt.c, apenas remova a linha #define _NO_PROTO do inicio do arquivo!

Se você estiver utilizando o compilador CC da Compac, a seguinte linha de configuração deverá funcionar:

```
CC="cc -pthread"
CFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host"
CXX="cxx -pthread"
CXXFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host \
-noexceptions -nortti"
export CC CFLAGS CXX CXXFLAGS
./configure \
--prefix=/usr/local/mysql \
--with-low-memory \
--enable-large-files \
--enable-shared=yes \
--with-named-thread-libs="-lpthread -lmach -lexc -lc"
gnumake
```

Se você tiver problemas com a libtool, ao compilar com bibliotecas compartilhadas como no exemplo acima, quando estiver ligando ao `mysqld`, você deve conseguir contornar este problema usando:

```
cd mysql
/bin/sh ../libtool --mode=link cxx -pthread  -O3 -DDBUG_OFF \
-O4 -ansi_alias -ansi_args -fast -inline speed \
-speculate all \ -arch host  -DUNDEF_HAVE_GETHOSTBYNAME_R \
-o mysql  mysql.o readline.o sql_string.o completion_hash.o \
../readline/libreadline.a -lcurses \
../libmysql/.libs/libmysqlclient.so  -lm
cd ..
gnumake
gnumake install
scripts/mysql_install_db
```

2.6.6.7 Notas Alpha-DEC-OSF1

Se você tiver problemas com compilação e tem o DEC `CC` e o `gcc` instalados, tente executar o `configure` desta forma:

```
CC=cc CFLAGS=-O CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Se você tiver problemas com o arquivo '`c_asm.h`', você pode criar e usar um arquivo '`c_asm.h`' 'burro' com:

```
touch include/c_asm.h
CC=gcc CFLAGS=-I./include \
CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Perceba que os seguintes problemas com o programa `ld` pode ser corrigido fazendo o download do último kit de atualização da DEC (Compaq) de `http://ftp.support.compaq.com/public/unix/`.

Com o OSF1 V4.0D e o compilador "DEC C V5.6-071 no Digital Unix V4.0 (Rev. 878)" o compilador tem alguns comportamentos estranhos (simbolos `asm` indefinidos). `/bin/ld` também aparece estar quebrado (problemas com erros `_exit undefined` ocorrendo ao ligar no `mysqld`). Neste sistema, temos compilado o MySQL com a seguinte linha `configure`, depois de substituir `/bin/ld` com a versão do OSF 4.0C:

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
```

Com o compilador da Digital "C++ V6.1-029", o seguinte deve funcionar:

```
CC=cc -pthread
CFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all \
       -arch host
CXX=cxx -pthread
CXXFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all \
          -arch host -noexceptions -nortti
export CC CFLAGS CXX CXXFLAGS
./configure --prefix=/usr/mysql/mysql --with-mysqld-ldflags=-all-static \
            --disable-shared --with-named-thread-libs="-lmach -lexc -lc"
```

Em algumas versões do OSF1, a função `alloca()` está quebrada. Corrija isto removendo a linha no '`config.h`' que define `'HAVE_ALLOCA'`.

A função `alloca()` pode também ter um protótipo incorreto em `/usr/include/alloca.h`. O alerta resultante deste erro pode ser ignorado.

`configure` irá usar a seguinte biblioteca thread automaticamente: `--with-named-thread-libs="-lpthread -lmach -lexc -lc"`.

Quando usar o `gcc`, você também pode tentar executar `configure` desta forma:

```
shell> CFLAGS=-D_PTHREAD_USE_D4 CXX=gcc CXXFLAGS=-O3 ./configure ....
```

Se você tiver problemas com sinais (MySQL finalzar inesperadamente sobre alta carga), você pode ter encontrado um erro com threads e sinais no SO. Neste caso você pode dizer ao MySQL para não usar sinais configurando-o com:

```
shell> CFLAGS=-DDONT_USE_THR_ALARM \
       CXXFLAGS=-DDONT_USE_THR_ALARM \
       ./configure ...
```

Isto não afeta a performance do MySQL, mas tem efeitos colaterais que não permitem finalizar clientes que estão "dormindo" em uma conexão com `mysqladmin kill` ou `mysqladmin shutdown`. Neste caso o cliente irá morrer quando ele receber o próximo comando.

Com `gcc` 2.95.2, você provavelmente encontrará o seguinte erro de compilação:

```
sql_acl.cc:1456: Internal compiler error in 'scan_region', at except.c:2566
Please submit a full bug report.
```

Para corrigir isto você deve alterar para o diretório `sql` e fazer um "corta e cola" da última linha `gcc`, mas altere `-O3` para `-O0` (ou adicione `-O0` imediatamente depois de `gcc` se você não tiver algumas opção `-O` na sua linha de compilação.) Depois disto feito você deve apenas voltar ao diretório superior e executar `make` novamente.

2.6.6.8 Notas SGI Irix

Se você estiver usando Irix Versão 6.5.3 ou mais novo, o `mysqld` só irá conseguir criar threads se você executá-lo como um usuário com privilégios de `CAP_SCHED_MGT` (como `root`) ou dar ao servidor `mysqld` este privilégio com o seguinte comando shell:

```
shell> chcap "CAP_SCHED_MGT+epi" /opt/mysql/libexec/mysqld
```

Você pode precisar indefinir alguns simbolos em '`config.h`' depois de executar `configure` e antes de compilar.

Em algumas implementações Irix, a função `alloca()` está quebrada. Se o servidor `mysqld` morrer em alguma instrução `SELECT`, remova as linhas de '`config.h`' que definem `HAVE_ALLOC` e `HAVE_ALLOC_H`. Se `mysqladmin create` não funciona, remova a linha do '`config.h`' que define `HAVE_READDIR_R`. Você também deve precisar remover a linha `HAVE_TERM_H`.

A SGI recomenda que você instale todos os patches desta página: http://support.sgi.com/surfzone/patches/

No mínimo, você deve instalar o último rollup do kernel, o último rollup `rld`, e o último rollup `libc`.

Definitivamente você precisará de todos patches POSIX nesta página, para suporte pthreads:

`http://support.sgi.com/surfzone/patches/patchset/6.2_posix.rps.html`

Se você obter o seguinte erro quando estiver compilando o '`mysql.cc`':

```
"/usr/include/curses.h", line 82: error(1084): invalid combination of type
```

Digite o seguinte no diretório topo da sua árvore fonte do MySQL:

```
shell> extra/replace bool curses_bool < /usr/include/curses.h \
> include/curses.h
shell> make
```

Existem relatos de problemas com organização de threads. Se somente uma thread estiver executando, o sistema fica lento. Pode se evitar isto iniciando outro cliente. Isto pode acarretar num crescimento de 2 para 10 vezes na velocidade de execução para a outra thread. Isto é um problema não compreendido com threads Irix; você deve improvisar para encontrar soluções até que isto seja resolvido.

Se você estiver compilando com `gcc`, você pode usar o seguinte comando `configure`:

```
CC=gcc CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql --enable-thread-safe-client \
--with-named-thread-libs=-lpthread
```

No Irix 6.5.11 com Irix C nativo e compiladores C++ ver. 7.3.1.2, o seguinte irá funcionar

```
CC=cc CXX=CC CFLAGS='-O3 -n32 -TARG:platform=IP22 -I/usr/local/include \
-L/usr/local/lib' CXXFLAGS='-O3 -n32 -TARG:platform=IP22 \
-I/usr/local/include -L/usr/local/lib' ./configure \
--prefix=/usr/local/mysql --with-innodb --with-berkeley-db \
--with-libwrap=/usr/local \
--with-named-curses-libs=/usr/local/lib/libncurses.a
```

2.6.6.9 Notas SCO

A versão atual foi testado somente nos sistemas "sco3.2v5.0.4" e "sco3.2v5.0.5". A versãoo para o "sco 3.2v4.2" também tem tido muito progresso.

Até o momento o compilador recomendado no OpenServer é o gcc 2.95.2. Com isto você deve estar apto a compilar o MySQL apenas com:

```
CC=gcc CXX=gcc ./configure ... (opções)
```

1. Para o OpenServer 5.0.X você precisa usar gcc-2.95.2p1 ou mais novo da Skunkware. `http://www.SCO.com/skunkware/` e ecolher o pacote OpenServer browser ou por ftp em ftp to ftp2.SCO.com no diretório pub/skunkware/osr5/devtools/gcc.
2. Você precisa do GCC versão 2.5.x para este produto e do sistema de desenvolvimento. Eles são necessários nesta versão do SCO Unix. Você não pode usar apenas o sistema GCC Dev.
3. Você deve obter o pacote FSU Pthreads e instalá-lo primeiro. Pode ser obtido em `http://www.cs.wustl.edu/~schmidt/ACE_wrappers/FSU-threads.tar.gz`. Você pode também obter um pacote precompilado de `http://www.mysql.com/Downloads/SCO/FSU-threads`
4. FSU Pthreads pode ser compilado com SCO Unix 4.2 com tcpip, ou OpenServer 3.0 ou OpenDesktop 3.0 (OS 3.0 ODT 3.0), com o Sistema de Desenvolvimento da SCO instalado usando uma boa versão do GCC 2.5.x ODT ou OS 3.0, no qual você necessitará de uma boa versão do GCC 2.5.x. Existem vários problemas sem uma boa versão. Esta versão do produto necessita do sistema de Desenvolvimento SCO Unix. Sem ele, você estará perdendo as bibliotecas e o editor de ligação necessário.
5. Para construir a FSU Pthreads no seu sistema, faça o seguinte:
   1. Execute `./configure` no diretório '`threads/src`' e selecione a opção SCO OpenServer. Este comando copia '`Makefile.SCO5`' para '`Makefile`'.
   2. Execute `make`.
   3. Para instalar no diretório padrão '`/usr/include`', use o usuário root, depois mude para o diretório '`thread/src`' e execute `make install`
6. Lembre de usar o GNU `make` quando estiver construindo o MySQL.
7. Se você não iniciou o `mysqld_safe` como root, você provavelmente só irá obter, por padrão, os 110 arquivos abertos por processo. O `mysqld` irá gravar uma nota sobre isto no arquivo log.
8. Com o SCO 3.2V5.0.5, você deve usar o FSU Pthreads versão 3.5c ou mais nova. Você deve também usar o gcc 2.95.2 ou mais novo.

   O seguinte comando `configure` deve funcionar:

   ```
   shell> ./configure --prefix=/usr/local/mysql --disable-shared
   ```

9. Com SCO 3.2V4.2, você deve usar FSU Pthreads versão 3.5c ou mais nova. O seguinte comando `configure` deve funcionar:

   ```
   shell> CFLAGS="-D_XOPEN_XPG4" CXX=gcc CXXFLAGS="-D_XOPEN_XPG4" \
          ./configure \
              --prefix=/usr/local/mysql \
              --with-named-thread-libs="-lgthreads -lsocket -lgen -lgthreads" \
              --with-named-curses-libs="-lcurses"
   ```

Você pode ter alguns problemas com alguns arquivos de inclusão. Neste caso, você pode encontrar novos arquivos de inclusão específicos do SCO em `http://www.mysql.com/Downloads/SCO/SCO-3.2v4.2-includes.tar.gz`. Você deve descompactar este arquivo no diretório '`include`' da sua árvore fonte do MySQL.

Notas de desenvolvimento SCO:

- O MySQL deve detectar automaticamente FSU Pthreads e ligar o `mysqld` com `-lgthreads -lsocket -lgthreads`.
- As bibliotecas de desenvolvimento SCO são re-entrantes nas FSU Pthreads. A SCO diz que suas bibliotecas de funções são re-entrantes, então elas devem ser re-entrantes com as FSU-Pthreads. FSU Pthreads no OpenServer tentam usar o esquema SCO para criar bibliotecas re-entrantes.
- FSU Pthreads (ao menos a versão em `http://www.mysql.com`) vem ligada com GNU `malloc`. Se você encontrar problemas com uso de memória, tenha certeza que o '`gmalloc.o`' esteja incluído em '`libgthreads.a`' e '`libgthreads.so`'.
- Na FSU Pthreads, as seguintes chamadas de sistema são compatíveis com pthreads: `read()`, `write()`, `getmsg()`, `connect()`, `accept()`, `select()` e `wait()`.
- O CSSA-2001-SCO.35.2 (O patch é listado de costume como patch de segurança erg711905-dscr_remap ver 2.0.0) quebra FSU threads e deixa o mysqld instável. Você deve remove-lo se você deseja executar o mysqld em uma máquina OpenServer 5.0.6.
- A SCO fornece Patches do Sistema Operacional em `ftp://ftp.sco.com/pub/openserver5` para OpenServer 5.0.x
- A SCO fornece correções de segurança e libsocket.so.2 em `ftp://ftp.sco.com/pub/security/OpenSer` e `ftp://ftp.sco.com/pub/security/sse` para OpenServer 5.0.x
- Correções de segurança pre-OSR506. Também a correção do telnetd em `ftp://stage.caldera.com/pub/security/openserver/` ou `ftp://stage.caldera.com/pub/securi` com a libsocket.so.2 e libresolv.so.1 com instruções para instalar em sistemas pre-OSR506.

  É provavelmente uma boa idéia para instalar os patches acima tentando compilar/usar o MySQL.

Se você deseja instalar o DBI no SCO, você deve editar o '`Makefile`' em DBI-xxx e cada subdiretório.

Note que o exemplo abaixo considera o gcc 2.95.2 ou mais novo:

```
OLD:                                    NEW:
CC = cc                                 CC = gcc
CCCDLFLAGS = -KPIC -W1,-Bexport         CCCDLFLAGS = -fpic
CCDLFLAGS = -wl,-Bexport                CCDLFLAGS =

LD = ld                                 LD = gcc -G -fpic
LDDLFLAGS = -G -L/usr/local/lib         LDDLFLAGS = -L/usr/local/lib
LDFLAGS = -belf -L/usr/local/lib        LDFLAGS = -L/usr/local/lib

LD = ld                                 LD = gcc -G -fpic
OPTIMISE = -Od                          OPTIMISE = -O1
```

```
OLD:
CCCFLAGS = -belf -dy -w0 -U M_XENIX -DPERL_SCO5 -I/usr/local/include

NEW:
CCFLAGS = -U M_XENIX -DPERL_SCO5 -I/usr/local/include
```

Isto é porque o carregador dinâmico Perl não irá carregar os módulos `DBI` se elas foram compiladas com `icc` ou `cc`.

Perl trabalha melhor quando compilado com `cc`.

2.6.6.10 Notas SCO Unixware Version 7.0

Você deve usar uma versão de MySQL pelo menos tão recente quando a Versão 3.22.13 e UnixWare 7.1.0 porque esta versão corrige alguns problemas de portabilidade sob o Unixware.

Nós temos compilado o MySQL com o seguinte comando `configure` no UnixWare Versão 7.1.x:

```
CC=cc CXX=CC ./configure --prefix=/usr/local/mysql
```

Se você deseja usar o `gcc`, deverá ser usado o `gcc` 2.95.2 ou mais novo.

```
CC=gcc CXX=g++ ./configure --prefix=/usr/local/mysql
```

1. A SCO fornece Patches do Sistema Operacional em `ftp://ftp.sco.com/pub/unixware7` para UnixWare 7.1.1 e 7.1.3 `ftp://ftp.sco.com/pub/openunix8` para OpenUNIX 8.0.0
2. A SCO fornece informação sobre Security Fixes em `ftp://ftp.sco.com/pub/security/OpenUNIX` para OpenUNIX `ftp://ftp.sco.com/pub/security/UnixWare` para UnixWare

2.6.7 Notas OS/2

O MySQL usa poucos arquivos aberto. Por isto, você deve adicionar uma linha parecida com a abaixo em seu arquivo '`CONFIG.SYS`':

```
SET EMXOPT=-c -n -h1024
```

Se você não fizer isto, provavelmente vai ter o seguinte erro:

```
File 'xxxx' not found (Errcode: 24)
```

Quando usar o MysQL com OS/2 Warp 3, o FixPack 29 ou superior é necessário. Com OS/2 Warp 4, FixPack 4 ou acima é necessário. Isto é uma exigência da biblioteca Pthreads. O MySQL deve estar instalado em uma partição que suporta nomes longos de arquivos como no HPFS, FAT32, etc.

O script '`INSTALL.CMD`' deve ser executado pelo próprio '`CMD.EXE`' do OS/2 e opde não funcionar com shells substitutas como o '`4OS2.EXE`'.

O script '`scripts/mysql-install-db`' foi renomeado. Agora ele é chamado '`install.cmd`' e é um script REXX, que irá atualizar as configurações padrões de segurança do MySQL e criar os ícones na WorkPlace Shell para o MySQL.

Suporte a módulos dinâmicos é compilado mas não totalmente testado. Módulos dinâmicos devem ser compilados usando a biblioteca run-time Pthreads.

```
gcc -Zdll -Zmt -Zcrtdll=pthrdrtl -I../include -I../regex -I.. \
    -o example udf_example.cc -L../lib -lmysqlclient udf_example.def
mv example.dll example.udf
```

Nota: Devido a limitações no OS/2, o nome do módulo UDF não deve esceder 8 caracteres. Módulos são armazenados no diretório '`/mysql2/udf`'; o script `safe-mysqld.cmd` irá colocar este diretório na variável de ambiente `BEGINLIBPATH`. Quando usando módulos UDF, extensões específicas são ignoradas — consuidera-se que seja '`.udf`'. Por exemplo, no Unix, o módulo compartilhado deve ser nomeado '`example.so`' e você deve carregar uma função dele desta forma:

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "example.so";
```

No OS/2, o módulo deve ter o nome de '`example.udf`', mas você não deve especificar a extensão do módulo:

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "example";
```

2.6.8 Notas Novell NetWare

Portar o `MySQL` para `NetWare` foi um grande esforço da `Novell`. Os clientes da Novell estarão satisfeitos ao notarem que o NetWare 6.5 virá com os binários do MySQL, completa com uma licença de uso comercial automatica para todos os servidores executando esta versão do NetWare.

Veja Seção 2.1.4 [Instalação NetWare], Página 74.

MySQL para NetWare é compilado usando um combinação do `Metrowerks CodeWarrior for NetWare` e uma versão especial de compilação cruzada do GNU autotools. Verifique esta seção no futuro para mais informações sobre construção e otimização do MySQL para NetWare.

2.6.9 Notas BeOS

Nós já falamos com alguns desenvolvedores BeOS que disseram que o MySQL está 80% portado para o BeOS, mas nós não sabemos qual a situação no momento.

2.7 Comentários de Instalação do Perl

2.7.1 Instalando Perl no Unix

O suporte Perl para o MySQL é fornecido pela interface cliente `DBI/DBD`. Veja Seção 12.5 [Perl], Página 881. O código do cliente Perl `DBD/DBI` exige Perl Versão 5.004 ou posterior. A interface **não funcionará** se você tiver uma versão mais do Perl.

O suporte MySQL Perl também exige que você tenha instalado o suporte a programação do cliente MySQL. Se você instalou o MySQL a partir de arquivos RPM, os programas cliente estão no cliente RPM, mas o suporte a programação do cliente está no RPM de desenvolvimento. Certifique de se instalar este RPM posteriormente.

Na Versão 3.22.8, o suporte Perl é distribuído separadamente do dsitribuição principal do MySQL. Se você quiser instalar o suporte Perl, os arquivos que você precisrá pode ser obtidos em `http://www.mysql.com/downloads/api-dbi.html`.

As distribuições Perl são fornecidas como arquios `tar` compactados e são chamados '`MODULE-VERSION.tar.gz`', onde `MODULE` é o nome do modulo e `VERSION` é o número da versão. Você deve conseguir as distribuições `Data-Dumper`, `DBI`, e `DBD-mysql` e instalá-las nesta ordem. O procedimento de instalação é mostrado aqui. O exemplo mostrado é para o módulo `Data-Dumper`, mas o procedimento é o mesmo para todas as distribuições:

1. Descompacte as distribuições no diretório atual:

   ```
   shell> gunzip < Data-Dumper-VERSION.tar.gz | tar xvf -
   ```

   Este comando cria um diretório chamado '`Data-Dumper-VERSION`'.
2. Entre no diretório principal da distribuição descompactada:

   ```
   shell> cd Data-Dumper-VERSION
   ```
3. Contrua a dsitribuição e compile tudo:

   ```
   shell> perl Makefile.PL
   shell> make
   shell> make test
   shell> make install
   ```

O comando `make test` é importante porque verifica que o módulo está funcionando. Note que ao executar este comando durante a instalação do `DBD-mysql` para exercitar o código da interface, o servidor MySQL deve estar em execução ou teste irá falhar.

É uma boa idéia reconstruir e reinstalar a distribuição `DBD-mysql` mesmo se você instalar uma nova distribuição do MySQL, particularmente se você notar simntomas como se todos os seus scripts `DBI` realizarem dump core depois de você atualizar o MySQL.

Se você não tem o direito para instalar os módulos Perl no diretório de sistema ou se você quiser instalar módulos Perl locais, a seguinte referência pode ajudá-lo:

`http://servers.digitaldaze.com/extensions/perl/modules.html#modules`

Procure sob o título `Installing New Modules that Require Locally Installed Modules`.

2.7.2 Instalaando ActiveState Perl no Windows

Para instalar o módulo `DBD` do MySQL com ActiveState Perl no Windows, você deve fazer o seguinte:

- Obter o ActiveState Perl em `http://www.activestate.com/Products/ActivePerl/` e instalá-lo.
- Abrir um prompt do DOS.
- Se exigido, configurar a variável `HTTP_proxy`. Por exemplo, você pode tentar:

  ```
  set HTTP_proxy=my.proxy.com:3128
  ```
- Inicie o progrma PPM:

  ```
  C:\> c:\perl\bin\ppm.pl
  ```
- Se você já não o fez, instale o `DBI`:

```
ppm> install DBI
```

- Se der tudo certo, execute o seguinte comando:

```
install \
ftp://ftp.de.uu.net/pub/CPAN/authors/id/JWIED/DBD-mysql-1.2212.x86.ppd
```

O acima deve funcionar pelo menos com o ActiveState Perl Versão 5.6.

Se você não puder fazer o mostrado acima funcionar, você deve instalar o driver `MyODBC` e conectar ao servidor MySQL através do ODBC:

```
use DBI;
$dbh= DBI->connect("DBI:ODBC:$dsn",$user,$password) ||
  die "Got error $DBI::errstr when connecting to $dsn\n";
```

2.7.3 Problemas Usando a Interface Perl `DBI/DBD`

Se Perl informar que não pode encontrar o módulo '`../mysql/mysql.so`', então o problema mais provável é que o Perl não pode localizar a biblioteca compartilhada '`libmysqlclient.so`'.

Você pode corrigir isto por qualquer um dos seguintes métodos:

- Compile a distribuição `DBD-mysql` com `perl Makefile.PL -static -config` em vez de `perl Makefile.PL`.
- Copie '`libmysqlclient.so`' para a diretório onde sua bibliotecas compartilhadas estão localizadas (provavelmente '`/usr/lib`' ou '`/lib`').
- No Linux você pode adicionar o caminho do diretório onde '`libmysqlclient.so`' está localizado ao arquivo '`/etc/ld.so.conf`'.
- Adicione o caminho do diretório onde '`libmysqlclient.so`' está localizada à variável de ambiente `LD_RUN_PATH`.

Se voce receber os seguintes erros de `DBD-mysql`, você provavelmente está usando `gcc` (ou usando um binário antigo compilado com `gcc`):

```
/usr/bin/perl: can't resolve symbol '__moddi3'
/usr/bin/perl: can't resolve symbol '__divdi3'
```

Adicione `-L/usr/lib/gcc-lib/... -lgcc` ao comando de ligação quando a biblioteca '`mysql.so`' estiver construída (verifique a saída de `make` para '`mysql.so`' quando você compilar o cliente Perl). A opção `-L` deve especificar o caminho do diretório onde '`libgcc.a`' está localizada no seu sistema.

Outra causa deste problema pode ser que Perl e o MySQL não são compilados com `gcc`. Neste caso, você pode resolver o problema compilando ambos com `gcc`.

Se você receber o seguinte erro de `DBD-mysql` quando executar o teste:

```
t/00base.....install_driver(mysql) failed:
Can't load '../blib/arch/auto/DBD/mysql/mysql.so' for module DBD::mysql:
../blib/arch/auto/DBD/mysql/mysql.so: undefined symbol:
uncompress at /usr/lib/perl5/5.00503/i586-linux/DynaLoader.pm line 169.
```

significa que você precisa adicionar a biblioteca compactada, `-lz`, a sua linha de ligação. Isto pode ser feito com a seguinte alteração no arquivo '`lib/DBD/mysql/Install.pm`':

```
$sysliblist .= " -lm";
```

Altere esta linha para:

```
$sysliblist .= " -lm -lz";
```

Depois disto, você **deve** executar 'make realclean' e proceder com o instalação desde o início.

Se você quiser usar o módulo Perl em um sistema que não suporta ligação dinâmica (como SCO) você pode gerar uma versão estática do Perl que inclui `DBI` e `DBD-mysql`. O modo que isto funciona é que você gera uma versão do Perl com o çodigo `DBI` ligado e instalado no topo do seu Perl atual. Entao você o utiliza para construir uma versão do Perl que adicionalmente tem o código `DBD` ligado em si, e instale-o.

No SCO, você deve ter as seguintes variáveis de ambiente configuradas:

```
shell> LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/usr/progressive/lib
ou
shell> LD_LIBRARY_PATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:\
/usr/progressive/lib:/usr/skunk/lib
shell> LIBPATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:\
/usr/progressive/lib:/usr/skunk/lib
shell> MANPATH=scohelp:/usr/man:/usr/local1/man:/usr/local/man:\
/usr/skunk/man:
```

Primeiro crie um Perl que inclui um módulo `DBI` ligado estaticamente executando estes comandos no diretório onde a sua distribuição `DBI` está localiada:

```
shell> perl Makefile.PL -static -config
shell> make
shell> make install
shell> make perl
```

Então você deve intalar o novo Perl. A saída de `make perl` indicará o comando `make` exato que você precisará executar para realizar a instalação. No SCO, isto é `make -f Makefile.aperl inst_perl MAP_TARGET=perl`.

A seguir use o Perl récem criado para criar outro Perl que também inclui uma `DBD::mysql` estaticamente ligado rodando estes comandos no diretório onde sua distribuição `DBD-mysql` está localizada:

```
shell> perl Makefile.PL -static -config
shell> make
shell> make install
shell> make perl
```

Finalmente você deve instalar este novo Perl. Novamente, a saída de `make perl` indica o comando a usar.

3 Tutorial de Introdução Do MySQL

Este capítulo fornece um tutorial de introdução ao MySQL demonstrando como usar o programa cliente `mysql` para criar e usar um banco de dados simples. `mysql` (algumas vezes apresentado como o "terminal monitor" ou apenas "monitor") é um programa interativo que lhe permite conectar a um servidor MySQL, executar consultas e visualizar os resultados. `mysql` pode também ser executado em modo batch: você coloca suas consultas em um arquivo, depois diz ao `mysql` para executar o conteúdo do arquivo. Cobrimos aqui ambas as formas de utilizar o `mysql`.

Para ver uma lista de opções conhecidas pelo `mysql`, chame-o com a opção `--help`:

```
shell> mysql --help
```

Este capítulo presume que o `mysql` está instalado na sua máquina e que um servidor MySQL está disponível para quem puder conectar. Se isto não for verdade, contate seu administrador MySQL. (Se *você* é o administrador, você precisará consultar outras seções deste manual.)

Este capítulo descreve todo o processo de configuração e uso de um banco de dados. Se você estiver interessado em apenas acessar um banco de dados já existente, podera pular as seções que descrevem como criar o banco de dados e suas respectivas tabelas.

Como este capítulo é um tutorial, vários detalhes são necessariamente omitidos. Consulte as seções relevantes do manual para mais informações sobre os tópicos cobertos aqui.

3.1 Conectando e Desconectando do Servidor

Para conectar ao servidor, normalmente você precisará fornecer um nome de usuário quando o `mysql` for chamado e, na maioria dos casos, uma senha. Se o servidor executa em uma máquina diferente de onde você está, você também precisará especificar um nome de máquina. Contate seu administrador para saber quais parâmetros de conexão você deve usar para conectar (isto é, qual máquina, usuário e senha usar). Uma vez que você saiba quais os parâmetros corretos, você deve estar pronto para conectar da seguinte forma:

```
shell> mysql -h servidor -u usuario -p
Enter password: ********
```

Os asteriscos (`********`) representam sua senha; digite-a quando o `mysql` mostrar o prompt `Enter password:`.

Se isto funcionar, você deve ver algumas informações iniciais seguidas de um prompt `mysql>`

```
shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25338 to server version: 4.0.14-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

O prompt lhe diz que o `mysql` está pronto para que você digite os comandos.

Algumas instalações MySQL permitem aos usuários de se conectarem como usuários anônimos ao servidor executando na máquina local. Se isto é o caso na sua máquina, você deve conseguir conectar ao servidor chamando o `mysql` sem qualquer opção:

```
shell> mysql
```

Depois de você conectar com sucesso, você pode disconectar a qualquer hora digitando `QUIT` (ou `\q`) no prompt `mysql>`:

```
mysql> QUIT
Bye
```

No Unix, você também pode desconectar pressionando Control-D.

A maioria dos exemplos nas seções seguintes assumem que você já está conectado ao servidor. Isto é indicado pelo prompt `mysql>`.

3.2 Fazendo Consultas

Tenha certeza que você está conectado ao servidor, como discutido na seção anterior. Isto feito, não será selecionado nenhum banco de dados para trabalhar, mas não tem problemas. Neste momento, é mais importante saber um pouco sobre como fazer consultas do que já criar tabelas, carregar dados para elas, e recuperar dados delas. Esta seção descreve os princípios básicos da entrada de comandos, usando diversas consultas você pode tentar se familiarizar com o funcionamento do `mysql`.

Aqui está um comando simples que solicita ao servidor seu número de versão e a data atual. Digite-o como visto abaixo seguindo o prompt `mysql>` e digite a tecla RETURN:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| version()    | CURRENT_DATE |
+--------------+--------------+
| 3.22.20a-log | 1999-03-19   |
+--------------+--------------+
1 row in set (0.01 sec)
mysql>
```

Esta consulta ilustra várias coisas sobre o `mysql`:

- Um comando normalmente consiste de uma instrução SQL seguida por um ponto e vírgula. (Existem algumas exceções onde um ponto e vírgula podem ser omitidos. `QUIT` mencionado anteriormente, é um deles. Saberemos de outros mais tarde.)
- Quando você emite um comando, o `mysql` o envia para o servidor para execução e mostra os resultados, depois imprime outro prompt `mysql>` para indicar que está pronto para outro comando.
- O `mysql` mostra a saída da consulta em forma tabular (linhas e colunas). A primeira linha contém rótulos para as colunas. As linhas seguintes são o resultado da consulta. Normalmente, rótulos de colunas são os nomes das colunas que você busca das tabelas do banco de dados. Se você está recuperando o valor de uma expressão no lugar de uma coluna de tabela (como no exemplo já visto), o `mysql` rotula a coluna usando a própria expressão.

- O `mysql` mostra quantas linhas foram retornadas e quanto tempo a consulta levou para executar, o que lhe dá uma vaga idéia da performance do servidor. Estes valores são impreciso porque eles representam tempo de relógio (Não tempo de CPU ou de máquina), e porque eles são afetados pelos fatores como a carga do servidor e latência de rede. (Para resumir, a linha "rows in set" não é mostrada nos exemplos seguintes deste capítulo.)

Palavras Chave podem ser entradas em qualquer caso de letra. As seguintes consultas são equivalentes:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Aqui está outra consulta. Ela demonstra que você pode usar o `mysql` como uma calculadora simples:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-------------+---------+
| SIN(PI()/4) | (4+1)*5 |
+-------------+---------+
|    0.707107 |      25 |
+-------------+---------+
```

As consultas mostradas até agora têm sido instruções relativamente pequenas, de uma linha. Você pode também entrar com múltiplas instruções em uma única linha. Basta finalizar cada uma com um ponto e vírgula:

```
mysql> SELECT VERSION(); SELECT NOW();
+--------------+
| VERSION()    |
+--------------+
| 3.22.20a-log |
+--------------+

+---------------------+
| NOW()               |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

Um comando não necessita estar todo em uma única linha, então comandos extensos que necessitam de várias linhas não são um problema. O `mysql` determina onde sua instrução termina através do ponto e vírgula terminador, e não pelo final da linha de entrada. (Em outras palavras, o `myqsl` aceita entradas de livre formato: Ele coleta linhas de entrada mas não as executa até chegar o ponto e vírgula.)

Aqui está uma instrução simples usando múltiplas linhas:

```
mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+--------------------+--------------+
```

```
| USER()             | CURRENT_DATE |
+--------------------+--------------+
| joesmith@localhost | 1999-03-18   |
+--------------------+--------------+
```

Neste exemplo, note como o prompt altera de `mysql>` para `->` depois de você entrar a primeira linha de uma consulta com múltiplas linhas. Isto é como o `mysql` indica que ainda não achou uma instrução completa e está esperando pelo resto. O prompt é seu amigo, porque ele fornece um retorno valioso. Se você usa este retorno, você sempre estará ciente do que o `mysql` está esperando.

Se você decidir que não deseja executar um comando que está no meio do processo de entrada, cancele-o digitando `\c`:

```
mysql> SELECT
    -> USER()
    -> \c
mysql>
```

Note o prompt aqui também. Ele troca para o `mysql>` depois de você digitar `\c`, fornecendo retorno para indicar que o `mysql` está pronto para um novo comando.

A seguinte tabela mostra cada dos prompts que você pode ver e resume o que ele significa sobre o estado em que o `mysql` se encontra:

Prompt	**Significado**
`mysql>`	Pronto para novo comando.
`->`	Esperando pela próxima linha de comando com múltiplas linhas.
`'>`	Esperando pela próxima linha, coletando uma string que comece com uma aspas simples (‘’’).
`">`	Esperando pela próxima linha, coletando uma string que comece com aspas duplas (‘"’).
`` `> ``	Esperando pela próxima linha, coletando uma string que comece com crase (‘`’).

É muito comum instruções multi-linhas ocorrerem por acidente quando você pretende publicar um comando em uma única linha, mas esquece o ponto e vírgula terminador. Neste caso,o `mysql` espera por mais entrada:

```
mysql> SELECT USER()
    ->
```

Se isto ocorrer com você (acha que entrou uma instrução mas a única resposta é um prompt `->`), o mais provável é que o `mysql` está esperando pelo ponto e vírgula. Se você não perceber o que o prompt está lhe dizendo, você pode parar por um tempo antes de entender o que precisa fazer. Entre com um ponto e vírgula para completar a instrução, e o `mysql` irá executá-la:

```
mysql> SELECT USER()
    -> ;
+--------------------+
| USER()             |
+--------------------+
| joesmith@localhost |
+--------------------+
```

O prompt `'>` e `">` ocorrem durante a coleta de strings. No MySQL, você pode escrever strings utilizando os caracteres ‘'’ ou ‘"’ (por exemplo, `'hello'` ou `"goodbye"`), e o `mysql` permite a entrada de strings que consomem múltiplas linhas. Quando você ver um prompt `'>` ou `">`, significa que você digitou uma linha contendo uma string que começa com um caracter de aspas ‘'’ ou ‘"’ mas ainda não entrou com a aspas que termina a string. Isto é bom se você realmente está entrando com uma string com múltiplas linhas, mas qual é a probalidade disto acontecer ? Não muita. Geralmente, os prompts `'>` e `">` indicam que você, por algum descuido, esqueceu algum caracter de aspas. Por exemplo:

```
mysql> SELECT * FROM minha_tabela WHERE nome = "Smith AND idade < 30;
    ">
```

Se você entrar esta sentença `SELECT`, apertar ENTER e esperar pelo resultado, nada irá acontecer. Em vez de se perguntar o porquê desta query demorar tanto tempo, perceba a pista fornecida pelo prompt `">`. Ele lhe diz que o `mysql` espera pelo resto de uma string não terminada. (Você ve o erro na declaração? Falta a segunda aspas na string `"Smith`.)

O que fazer neste ponto ? A coisa mais simples é cancelar o comando. Entretanto, você não pode simplesmente digitar `\c` neste caso, porque o `mysql` o intrerpreta como parte da string que está coletando! Digite o caracter de aspas para fechar (então o `mysql` sabe que você fechou a string), então digite `\c`:

```
mysql> SELECT * FROM minha_tabela WHERE nome = "Smith AND idade < 30;
    "> "\c
mysql>
```

O prompt volta para `mysql>`, indicando que o `mysql` está pronto para um novo comando.

O prompt `‘>` é similar aos prompts `'>` e `">`, mas indica que você começou mas não completou um identificados citado com o sinal de crase.

É importante saber o que os prompts `'>`, `">` e `‘>` significam, porque se você entrar sem querer com uma string sem terminação, quaisquer linhas seguintes que forem digitadas serão ignoradas pelo `mysql` — incluindo uma linha contendo `QUIT`! Isto pode ser um pouco confuso, especialmente se você não sabe que você precisa fornecer as aspas finais antes poder cancelar o comando atual.

3.3 Criação e Utilização de um Banco de Dados

Agora que você já sabe como entrar com os comandos, é hora de acessar um banco de dados.

Suponha que você tenha diversos animais de estimação em sua casa (menagerie) e você gostaria de ter o registro de vários tipos de informações sobre eles. Você pode fazer isto criando tabelas para armazenar seus dados e carregá-los com a informação desejada. Depois você pode responder diferentes tipos de questões sobre seus animais recuperando dados das tabelas. Esta seção mostrará como:

- Criar um banco de dados
- Criar uma tabela
- Carregar dados na tabela
- Recuperar dados de uma tabela de várias maneiras
- Usar múltiplas tabelas

O banco de dados menagerie será simples (deliberadamente), mas não é difícil pensar em situações na vida real em que um tipo similar de banco de dados pode ser usado. Por exemplo, um banco de dados deste tipo pode ser usado por um fazendeiro para gerenciar seu estoque de animais, ou por um veterinário para gerenciar registros de seus pacientes. Uma distribuição do menagerie contendo algumas das consultas e dados de exemplos usados nas seções seguintes podem ser obtidas do site Web do MySQL. Estão disponíveis tanto no formato `tar` comprimido (`http://www.mysql.com/Downloads/Contrib/Examples/menagerie.tar.gz`) como no formato Zip (`http://www.mysql.com/Downloads/Contrib/Examples/menagerie.zip`).

Utilize a instrução `SHOW` para saber quais bancos de dados existem atualmente no servidor:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

A lista de bancos de dados provavelmente será diferente na sua máquina, mas os bancos de dados `mysql` e `test` provavelmente estarão entre eles. O banco de dados `mysql` é necessário porque ele descreve privilégios de acessos de usuários. O banco de dados `test` é geralamente fornecido como um espaço para que os usuários possam fazer testes.

Note que você não pode ver todos os banco de dados se você nãi tiver o privilégio `SHOW DATABASES`. Veja Seção 4.4.1 [GRANT], Página 254.

Se o banco de dados `test` existir, tente acessá-lo:

```
mysql> USE test
Database changed
```

Perceba que o `USE`, como o `QUIT`, não necessitam de um ponto e vírgula. (Você pode terminar tais declarações com uma ponto e vírgula se gostar; isto não importa) A instrução `USE` é especial em outra maneira, também: Ela deve ser usada em uma única linha.

Você opde usar o banco de dados `test` (Se você tiver acesso a ele) para os exemplos que seguem mas qualquer coisa que você criar neste banco de dados pode ser removido por qualquer um com acesso a ele. Por esta razão, você provavelmente deve pedir permissão ao seu administrador MySQL para usar um banco de dados próprio. Suponha que você o chame de `menagerie`. O administrador precisar executar um comando como este:

```
mysql> GRANT ALL ON menagerie.* TO 'your_mysql_name'@'your_client_host';
```

onde `seu_usuário_mysql` é o nome do usuário MySQL atribuido a você e `your_client_host` é a máquina da qual você se conecta ao servidor.

3.3.1 Criando e Selecionando um Banco de Dados

Se o administrador criar seu banco de dados quando configurar as suas permissões, você pode começar a usá-lo. Senão, você mesmo precisa criá-lo:

```
mysql> CREATE DATABASE menagerie;
```

No Unix, nomes de bancos de dados são caso sensitivo (ao contrário das palavras chave SQL), portanto você deve sempre fazer referência ao seu banco de dados como `menagerie` e não `Menagerie`, `MENAGERIE` ou outra variação. Isto também é verdade para nomes de tabelas. (No Windows, esta restrição não se aplica, entiretanto você deve referenciar os bancos de dados e tabelas usando o mesmo caso em toda a parte da consulta.)

Criar um bancos de dados não o seleciona para o uso; você deve fazer isso de forma explícita. Para fazer o `menagerie` o banco de dados atual, use o comando:

```
mysql> USE menagerie
Database changed
```

Seu banco de dados necessita ser criado somente uma única vez, mas você deve selecioná-lo para o uso cada vez que você iniciar uma seção `mysql`. Você pode fazer isso usando a instrução `USE` como visto no exemplo. Uma forma alternativa é selecionar o banco de dados na linha de comando quando você chamar o `mysql`. Apenas especifique seu nome depois de qualquer parâmetro de conexão que você pode precisar fornecer. Por exemplo:

```
shell> mysql -h servidor -u usuario -p menagerie
Enter password: ********
```

Perceba que `menagerie` não é sua senha no comando mostrado. Se você precisar passar sua senha na linha de comando depois da opção `-p`, você deve fazê-lo sem usar espaços (por exemplo, `-pminhasenha` e não como em `-p minhasenha`). Entretando, colocando sua senha na linha de comando não é recomendado, porque isto expõe sua senha permitindo que outro usuário utilize a sua máquina.

3.3.2 Criando uma Tabela

Criar o banco de dados é a parte fácil, mas neste ponto ele está vazio, como o `SHOW TABLES` mostrará:

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

A parte mais difícil é decidir qual a estrutura que seu banco de dados deve ter: quais tabelas você precisará e que colunas estarão em cada uma delas.

Você irá precisar de uma tabela para guardar um registro para cada um de seus animais de estimação. Esta tabela pode ser chamada `pet`, e ela deve conter, pelo menos, o nome de cada animal. Como o nome por si só não é muito interessante, a tabela deverá conter outras informações. Por exemplo, se mais de uma pessoa na sua família também tem animais, você pode desejar listar cada dono. Você pode também desejargravar algumas informações descritivas básicas como espécie e sexo.

Que tal a idade? Pode ser do interesse, mas não é uma boa coisa para se armazenar em um banco de dados. A idade muda à medida em que o tempo passa, o que significa que você sempre terá de atualizar seus registros. Em vez disso, é melhor armazenar um valor fixo como a data de nascimento. Então, sempre que você precisar da idade, basta você calculá-la como a diferença entre a data atual e a data de aniversário. O MySQL fornece funções para fazer aritmética de datas, então isto não é difícil. Armazenando datas de aniversário no lugar da idade também oferece outras vantagens:

- Você pode usar o banco de dados para tarefas como gerar lembretes para aniversários que estão chegando. (Se você pensa que este tipo de query é algo bobo, perceba que é a mesma questão que você perguntar no contexto de um banco de dados comercial para identificar clientes para quais você precisará enviar cartão de aniversário, para um toque pessoal assistido pelo computador.)
- Você pode calcular a idade em relação a outras datas diferente da data atual. Por exemplo, se você armazenar a data da morte no banco de dados, você poderá facilmente calcular qual a idade que o bicho tinha quando morreu.

Você provavelmente pode pensar em outros tipos de informações que poderão ser úteis na tabela `pet`, mas as identificadas até o momento são suficientes por agora: nome(name), dono(owner), espécie(species), sexo(sex), data de nascimento(birth) e data da morte(death).

Utilize a senteça `CREATE TABLE` para especificar o layout de sua tabela:

```
mysql> CREATE TABLE pet (nome VARCHAR(20), owner VARCHAR(20),
    -> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
```

`VARCHAR` é uma boa escolha para os campos `name`, `owner`, e `species` porque os valores da coluna são de tamanho variável. Os tamanhos destas colunas não precisam necessáriamente de ser os mesmos e não precisam ser `20`. Você pode escolher qualquer tamanho de `1` a `255`, o que você achar melhor. (Se você não fizer uma boa escolha e depois precisar de um campo maior, o MySQL fornece o comando `ALTER TABLE`.)

O sexo dos animais podem ser representados em várias formas, por exemplo, `"m"` e `"f"` ou mesmo `"macho"` e `"fêmea"`. É mais simples usar os caracteres `"m"` e `"f"`.

O uso do tipo de dados `DATE` para as colunas `birth` e `death` são obviamente a melhor escolha.

Agora que você criou uma tabela, a instrução `SHOW TABLES` deve produzir alguma saída:

```
mysql> SHOW TABLES;
+----------------------+
| Tables in menagerie |
+----------------------+
| pet                  |
+----------------------+
```

Para verificar se sua tabela foi criada da forma que você esperava, utilize a instrução `DESCRIBE`:

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

Você pode usar `DESCRIBE` a qualquer hora, por exemplo, se você esquecer os nomes das colunas na sua tabela ou de que tipos elas têm.

3.3.3 Carregando dados em uma tabela

Depois de criar sua tabela, você precisará povoá-la. As instruções `LOAD DATA` e `INSERT` são úteis para isto.

Suponha que seu registro de animais possa ser descrito como é abaixo: (Observe que o MySQL espera datas no formato `AAAA-MM-DD`; isto pode ser diferente do que você está acostumado.)

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	

Como você está começando com uma tabela vazia, uma forma simples de povoá-la é criar um arquivo texto contendo uma linha para cada um de seus animais, e depois carregar o conteúdo do arquivo para a tabela com uma simples instrução.

Você pode criar um arquivo texto '`pet.txt`' contendo um registro por linha, com valores separado por tabulações e na mesma ordem em que as colunas foram listadas na instrução `CREATE TABLE`. Para valores em falta (como sexo desconhecido ou data da morte para animais que ainda estão vivos), você pode usar valores `NULL`. Para representá-lo em seu arquivo texto, use `\N` (barra invertidam N maíusculo). Por exemplo, o registro para Whistler the bird podem parecer com isto (onde o espaço em branco entre os valores é um simples caractere de tabulação):

name	owner	species	sex	birth	death
Whistler	Gwen	bird	\N	1997-12-09	\N

Para carregar o arquivo texto '`pet.txt`' na tabela `pet`, use este comando:

```
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

Você pode especificar o valor do separador de colunas e o marcador de final de linha explicitamente na instrução `LOAD DATA` se você desejar. Mas os valores omitidos são suficientes para a instrução ler o arquivo '`pet.txt`' corretamente.

Se a instrução falhar, é desejável que a sua instalação do MySQL não tenha a capacidade do arquivo local habilitada por padrão. Veja Seção 4.3.4 [`LOAD DATA LOCAL`], Página 231 para informações sobre como alterar isto.

Quando você desejar adicionar novos registros um a um, a instrução `INSERT` é usada. Na sua forma mais simples, você fornece valores para cada coluna, na ordem em que as colunas foram listadas na instrução `CREATE TABLE`. Suponha que Diane tenha um novo hamster chamado Puffball. Você pode adicionar um registro utilizando uma instrução `INSERT` desta forma:

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

Perceba que os valores de string e datas são especificados aqui como strings com aspas. Com o `INSERT` você também pode inserir `NULL` diretamente para representar um valor em falta. Não pode ser usado `\N` como você fez com `LOAD DATA`.

A partir deste exemplo, você deverá perceber que existem várias outras formas envolvidas para carregar seus registros inicialmente utilizando diversas instruções `INSERT` do que uma simples instrução `LOAD DATA`.

3.3.4 Recuperando Informações de uma Tabela

A instrução `SELECT` é usada para recuperar informações de uma tabela. A forma geral da instrução é:

```
SELECT o_que_mostrar
FROM de_qual_tabela
WHERE condições_para_satisfazer;
```

`o_que_mostrar` indica o que você deseja ver. Isto pode ser uma lista de colunas ou `*` para indicar "todas colunas." `de_qual_tabela` indica a tabela de onde você deseja recuperar os dados. A cláusula `WHERE` é opcional. Se estiver presente, `condições_para_satisfazer` especificam as condições que os registros devem satisfazer para fazer parte do resultado.

3.3.4.1 Selecionando Todos os Dados

A forma mais simples do `SELECT` recuperar tudo de uma tabela:

```
mysql> SELECT * FROM pet;
+----------+--------+---------+------+------------+------------+
| name     | owner  | species | sex  | birth      | death      |
+----------+--------+---------+------+------------+------------+
| Fluffy   | Harold | cat     | f    | 1993-02-04 | NULL       |
| Claws    | Gwen   | cat     | m    | 1994-03-17 | NULL       |
| Buffy    | Harold | dog     | f    | 1989-05-13 | NULL       |
| Fang     | Benny  | dog     | m    | 1990-08-27 | NULL       |
| Bowser   | Diane  | dog     | m    | 1979-08-31 | 1995-07-29 |
| Chirpy   | Gwen   | bird    | f    | 1998-09-11 | NULL       |
| Whistler | Gwen   | bird    | NULL | 1997-12-09 | NULL       |
| Slim     | Benny  | snake   | m    | 1996-04-29 | NULL       |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Esta forma do `SELECT` é útil se você deseja ver sua tabela inteira como agora, depois de você acabar de carregá-la com os dados iniciais. Por exempo, você pode pensar que a data de nascimento do Bowser não está correta. Consultando seus papéis originais de pedigree, descobriu que o ano correto do nascimento deve ser 1989, não 1979.

Existem pelo menos duas formas de corrigir isto:

- Edite o arquivo '`pet.txt`' para corrigir o erro, depois limpe a tabela e recarregue-o usando `DELETE` e `LOAD DATA`:

  ```
  mysql> DELETE FROM pet;
  mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
  ```

Entretanto, se você fizer isto, você também deve refazer a entrada para Puffball.

- Corrigir somente o registro errado com uma instrução `UPDATE`:

```
mysql> UPDATE pet SET birth = "1989-08-31" WHERE name = "Bowser";
```

O `UPDATE` altera apenas o registro em questão e não exige que você recarregue a tabela.

3.3.4.2 Selecionando Registros Específicos

Como foi mostrado na seção anterior, é fácil recuperar uma tabela inteira. Apenas omita a cláusula `WHERE` da instrução `SELECT`. Mas normalmente você não quer ver toda a tabela, particularmente quando a tabela ficar grande. Em vez disso, você estará mais interessado em ter a resposta de uma questão em particular, no qual você especifica detalhes da informação que deseja. Vamos ver algumas consultas de seleção nos termos das questões sobre seus animais.

Você pode selecionar apenas registros específicos da sua tabela. Por exemplo, se você deseja verificar a alteração que fez na data de nascimento do Bowser, selecione o registro desta forma:

```
mysql> SELECT * FROM pet WHERE name = "Bowser";
+--------+-------+---------+------+------------+------------+
| name   | owner | species | sex  | birth      | death      |
+--------+-------+---------+------+------------+------------+
| Bowser | Diane | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+-------+---------+------+------------+------------+
```

A saída confirma que o ano foi gravado corretamente agora como 1989 e não 1979.

Comparações de strings normalmente são caso insensitivo, então você pode especificar o nome como `"bowser"`, `"BOWSER"`, etc. O resultado da pesquisa será o mesmo.

Você pode especificar condições em qualquer coluna, não apenas no `name`. Por exemplo, se você deseja saber quais foram os animais que nasceram depois de 1998, teste o campo `birth`:

```
mysql> SELECT * FROM pet WHERE birth >= "1998-1-1";
+----------+-------+---------+------+------------+-------+
| name     | owner | species | sex  | birth      | death |
+----------+-------+---------+------+------------+-------+
| Chirpy   | Gwen  | bird    | f    | 1998-09-11 | NULL  |
| Puffball | Diane | hamster | f    | 1999-03-30 | NULL  |
+----------+-------+---------+------+------------+-------+
```

Você pode combinar condições, por exemplo, para encontrar cadelas (dog/f):

```
mysql> SELECT * FROM pet WHERE species = "dog" AND sex = "f";
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

A consulta anterior utiliza o operador lógico `AND` (e). Existe também um operador `OR` (ou):

```
mysql> SELECT * FROM pet WHERE species = "snake" OR species = "bird";
+----------+-------+---------+------+------------+-------+
| name     | owner | species | sex  | birth      | death |
+----------+-------+---------+------+------------+-------+
| Chirpy   | Gwen  | bird    | f    | 1998-09-11 | NULL  |
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL  |
| Slim     | Benny | snake   | m    | 1996-04-29 | NULL  |
+----------+-------+---------+------+------------+-------+
```

`AND` e `OR` podem ser misturados, embora `AND` tem maior precedência que `OR`. Se você usar ambos os operadores, é uma ótima idéia usar parênteses para indicar explicitamente quais condições devem ser agrupadas:

```
mysql> SELECT * FROM pet WHERE (species = "cat" AND sex = "m")
    -> OR (species = "dog" AND sex = "f");
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

3.3.4.3 Selecionando Colunas Específicas

Se você não desejar ver todo o registro de sua tabela, especifique as colunas em que você estiver interessado, separado por vírgulas. Por exemplo, se você deseja saber quando seus animais nasceram, selecione as colunas `name` e `birth`:

```
mysql> SELECT name, birth FROM pet;
+----------+------------+
| name     | birth      |
+----------+------------+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Para saber quem são os donos dos animais, use esta consulta:

```
mysql> SELECT owner FROM pet;
+--------+
| owner  |
+--------+
| Harold |
```

```
| Gwen   |
| Harold |
| Benny  |
| Diane  |
| Gwen   |
| Gwen   |
| Benny  |
| Diane  |
+--------+
```

Entretanto, perceba que a query simplesmente retornou o campo `owner` de cada registro, e alguns deles apareceram mais de uma vez. Para minimizar a saída, recupere cada registro apenas uma vez, adicionando a palavra chave `DISTINCT`:

```
mysql> SELECT DISTINCT owner FROM pet;
+--------+
| owner  |
+--------+
| Benny  |
| Diane  |
| Gwen   |
| Harold |
+--------+
```

Você pode usar uma cláusula `WHERE` para combinar seleção de registros com seleção de colunas. Por exemplo, para obter a data de nascimento somente dos gatos e cachorros, utilize esta query:

```
mysql> SELECT name, species, birth FROM pet
    -> WHERE species = "dog" OR species = "cat";
+--------+---------+------------+
| name   | species | birth      |
+--------+---------+------------+
| Fluffy | cat     | 1993-02-04 |
| Claws  | cat     | 1994-03-17 |
| Buffy  | dog     | 1989-05-13 |
| Fang   | dog     | 1990-08-27 |
| Bowser | dog     | 1989-08-31 |
+--------+---------+------------+
```

3.3.4.4 Ordenando Registros

Você deve ter percebido nos exemplos anteriores que os registros retornados não são mostrados de forma ordenada. Normalmente é mais fácil examinar a saída da consulta quando os registros são ordenados com algum sentido. Para ordenar o resultado, utilize uma cláusula `ORDER BY`.

Aqui está o dia de nascimento dos animais, ordenado por data:

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
+----------+------------+
```

```
| name     | birth      |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Em colunas de tipo de caracter, ordenaição como qualquer outra operação de comparação é normalmente realizada no modo caso insensitivo. Isto significa que a ordem será indefinida para colunas que são idênticas exceto quanto ao caso da letra. Você pode forçar uma ordenação em caso senitivo para uma coluna usando a coerção BINARY: `ORDER BY BINARY(campo)`.

A ordenação padrão é crescente, com os valores menores em primeiro. Para ordenação na ordem reversa, adicione a palavra chave `DESC` (descendente) ao nome da coluna que deve ser ordenada:

```
mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
+----------+------------+
| name     | birth      |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

Você pode ordenar por múltiplas colunas e você pode classificar colunas em direções diferentes. Por exemplo, para ordenar o tipo de animal em ordem crescente, depois por dia de nascimento dentro do tipo de animal em ordem decrescente (com os mais novos primeiro), utilize a seguinte consulta:

```
mysql> SELECT name, species, birth FROM pet ORDER BY species, birth DESC;
+----------+---------+------------+
| name     | species | birth      |
+----------+---------+------------+
| Chirpy   | bird    | 1998-09-11 |
| Whistler | bird    | 1997-12-09 |
| Claws    | cat     | 1994-03-17 |
| Fluffy   | cat     | 1993-02-04 |
```

```
| Fang     | dog     | 1990-08-27 |
| Bowser   | dog     | 1989-08-31 |
| Buffy    | dog     | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim     | snake   | 1996-04-29 |
+----------+---------+------------+
```

Perceba que a palavra chave `DESC` aplica somente para o nome da coluna precedente (`birth`); ela não afeta a ordenação da coluna `species`.

3.3.4.5 Cálculo de Datas

O MySQL fornece várias funções que você pode usar para realizar cálculos em datas, por exemplo, para calcular idades ou extrair partes de datas.

Para determinar quantos anos cada um do seus animais tem, compute a diferença do ano da data atual e a data de nascimento (birth), depois subtraia se a o dia/mês da data atual for anterior ao dia/mês da data de nascimento. A consulta seguinte, mostra, para cada animal, a data de nascimento, a data atual e a idade em anos.

```
mysql> SELECT name, birth, CURDATE(),
    -> (YEAR(CURDATE())-YEAR(birth))
    -> - (RIGHT(CURDATE(),5)<RIGHT(birth,5))
    -> AS age
    -> FROM pet;
+----------+------------+------------+------+
| name     | birth      | CURDATE()  | age  |
+----------+------------+------------+------+
| Fluffy   | 1993-02-04 | 2003-08-19 |   10 |
| Claws    | 1994-03-17 | 2003-08-19 |    9 |
| Buffy    | 1989-05-13 | 2003-08-19 |   14 |
| Fang     | 1990-08-27 | 2003-08-19 |   12 |
| Bowser   | 1989-08-31 | 2003-08-19 |   13 |
| Chirpy   | 1998-09-11 | 2003-08-19 |    4 |
| Whistler | 1997-12-09 | 2003-08-19 |    5 |
| Slim     | 1996-04-29 | 2003-08-19 |    7 |
| Puffball | 1999-03-30 | 2003-08-19 |    4 |
+----------+------------+------------+------+
```

Aqui, `YEAR()` separa a parte do ano de uma data e `RIGHT()` separa os cinco caracteres mais a direita que representam a parte da data `MM-DD`. A parte da expressão que compara os valores `MM-DD` resulta em 1 ou 0, o qual ajusta a diferença do ano um ano abaixo se `CURDATE` ocorrer mais cedo, no ano, que `birth`. A expressão completa é um tanto deselegante, então um apelido (`age`) é usado para obter uma saída mais significativa.

A consulta funciona, mas o resultado pode ser mais compreensível se os registros forem apresentados em alguma ordem. Isto pode ser feito adicionando uma cláusula `ORDER BY name` para ordenar a saída pelo nome:

```
mysql> SELECT name, birth, CURDATE(),
```

```
    -> (YEAR(CURDATE())-YEAR(birth))
    -> - (RIGHT(CURDATE(),5)<RIGHT(birth,5))
    -> AS age
    -> FROM pet ORDER BY name;
+----------+------------+------------+------+
| name     | birth      | CURDATE()  | age  |
+----------+------------+------------+------+
| Bowser   | 1989-08-31 | 2003-08-19 |   13 |
| Buffy    | 1989-05-13 | 2003-08-19 |   14 |
| Chirpy   | 1998-09-11 | 2003-08-19 |    4 |
| Claws    | 1994-03-17 | 2003-08-19 |    9 |
| Fang     | 1990-08-27 | 2003-08-19 |   12 |
| Fluffy   | 1993-02-04 | 2003-08-19 |   10 |
| Puffball | 1999-03-30 | 2003-08-19 |    4 |
| Slim     | 1996-04-29 | 2003-08-19 |    7 |
| Whistler | 1997-12-09 | 2003-08-19 |    5 |
+----------+------------+------------+------+
```

Para ordenar a saída por `age` em vez de `name`, é só utilizar uma cláusua `ORDER BY` diferente:

```
mysql> SELECT name, birth, CURDATE(),
    -> (YEAR(CURDATE())-YEAR(birth))
    -> - (RIGHT(CURDATE(),5)<RIGHT(birth,5))
    -> AS age
    -> FROM pet ORDER BY age;
+----------+------------+------------+------+
| name     | birth      | CURDATE()  | age  |
+----------+------------+------------+------+
| Chirpy   | 1998-09-11 | 2003-08-19 |    4 |
| Puffball | 1999-03-30 | 2003-08-19 |    4 |
| Whistler | 1997-12-09 | 2003-08-19 |    5 |
| Slim     | 1996-04-29 | 2003-08-19 |    7 |
| Claws    | 1994-03-17 | 2003-08-19 |    9 |
| Fluffy   | 1993-02-04 | 2003-08-19 |   10 |
| Fang     | 1990-08-27 | 2003-08-19 |   12 |
| Bowser   | 1989-08-31 | 2003-08-19 |   13 |
| Buffy    | 1989-05-13 | 2003-08-19 |   14 |
+----------+------------+------------+------+
```

Uma consulta similar pode ser usada para determinar a idade na morte para animais que morreram. Para determinar quais são os animais, confira se o valor de `death` não é `NULL`. Depois para estes com valores não-`NULL`, compute a diferença entre os valores dos campos `death` e `birth`:

```
mysql> SELECT name, birth, death,
    -> (YEAR(death)-YEAR(birth)) - (RIGHT(death,5)<RIGHT(birth,5))
    -> AS age
    -> FROM pet WHERE death IS NOT NULL ORDER BY age;
+--------+------------+------------+------+
| name   | birth      | death      | age  |
```

```
+--------+------------+------------+------+
| Bowser | 1989-08-31 | 1995-07-29 |    5 |
+--------+------------+------------+------+
```

A consulta usa `death IS NOT NULL` em vez de `death != NULL` porque `NULL` é um valor especial que não pode ser comparada usando operadores comuns de comparação. Isto será explicado depois. Veja Seção 3.3.4.6 [Working with NULL], Página 185.

E se você desejar saber quais animais fazem aniversário no próximo mês? Para este tipo de cálculo, ano e dia são irrelevantes; você simplesmente deseja extrair a parte do mês da coluna `birth`. O MySQL fornece diversas funções para extrair partes da data, como em `YEAR()`, `MONTH()` e `DAYOFMONTH()`. `MONTH` é a função apropriada aqui. Para ver como ela funciona, execute uma consulta simples que mostre o valor de `birth` e `MONTH(birth)`:

```
mysql> SELECT name, birth, MONTH(birth) FROM pet;
+----------+------------+--------------+
| name     | birth      | MONTH(birth) |
+----------+------------+--------------+
| Fluffy   | 1993-02-04 |            2 |
| Claws    | 1994-03-17 |            3 |
| Buffy    | 1989-05-13 |            5 |
| Fang     | 1990-08-27 |            8 |
| Bowser   | 1989-08-31 |            8 |
| Chirpy   | 1998-09-11 |            9 |
| Whistler | 1997-12-09 |           12 |
| Slim     | 1996-04-29 |            4 |
| Puffball | 1999-03-30 |            3 |
+----------+------------+--------------+
```

Encontrar animais com aníversário no próximo mês também é fácil. Suponha que o mês atual é abril. Então o valor do mês é `4` e você procura por animais nascidos em Maio (mês `5`) assim:

```
mysql> SELECT name, birth FROM pet WHERE MONTH(birth) = 5;
+-------+------------+
| name  | birth      |
+-------+------------+
| Buffy | 1989-05-13 |
+-------+------------+
```

Existe uma pequena complicação se o mês atual é Dezembro, é claro. Você não pode apenas adicionar um para o número do mês (`12`) e procurar por animais nascidos no mês `13`, porque não existe tal mês. O certo seria procurar por animais nascidos em Janeiro (mês `1`).

Você pode também escrever uma consulta para que funcione sem importar qual é o mês atual. Assim você não têm quee usar um número de mês em particular na consulta. `DATE_ADD()` permite adicionar um intervalo de tempo para uma data fornecida. Se você adicionar um mês para o valor de `CURDATE`, então extrair a parte do mês com `MONTH()`, o resultado é o mês no qual você deseja procurar por aniversários:

```
mysql> SELECT name, birth FROM pet
    -> WHERE MONTH(birth) = MONTH(DATE_ADD(CURDATE(), INTERVAL 1 MONTH));
```

Uma maneira diferente para realizar a mesma tarefa é adicionar `1` para obter o mês seguinte ao atual (depois de usar a função módulo (`MOD`) para o valor do mês retornar `0` se ele for `12`):

```
mysql> SELECT name, birth FROM pet
    -> WHERE MONTH(birth) = MOD(MONTH(CURDATE()), 12) + 1;
```

Perceba que `MONTH` retorna um número entre `1` e `12`. E `MOD(alguma_coisa,12)` retorna um número entre `0` e `11`. Então a adição tem que ser feita depois do `MOD()`, senão iríamos de Novembro (`11`) para Janeiro (`1`).

3.3.4.6 Trabalhando com Valores Nulos (`NULL`)

O valor `NULL` pode ser supreendente até você usá-lo. Conceitualmente, `NULL` significa valor em falta ou valor desconhecido e é tratado de uma forma diferente de outros valores. Para testar o valor `NULL`, você não pode usar os operadores de comparações aritméticas como em `=`, `<`, ou `!=`. Para demonstrar para você mesmo, tente executar a seguinte consulta:

```
mysql> SELECT 1 = NULL, 1 != NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 != NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
|     NULL |      NULL |     NULL |     NULL |
+----------+-----------+----------+----------+
```

Claramente você não obterá resultados significativos destas comparações. Utilize os operadores `IS NULL` e `IS NOT NULL` no lugar:

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----------+---------------+
| 1 IS NULL | 1 IS NOT NULL |
+-----------+---------------+
|         0 |             1 |
+-----------+---------------+
```

No MySQL, `0` ou `NULL` significa falso e o resto é verdadeiro. O valor verdadeiro por o padrão em uma operação booleana é `1`.

Este tratamento especial de `NULL` é porque, na seção anterior, foi necessário determinar quais animais não estavam mais vivos usando `death IS NOT NULL` no lugar de `death <> NULL`.

Dois valores `NULL` são considerados como iguais em um `GROUP BY`.

Ao fazer um `ORDER BY`, valores `NULL` são apresentados primeiro se você fizer `ORDER BY ... ASC` e por último se você fizer `ORDER BY ... DESC`.

Note que o MySQL 4.0.2 a 4.0.10 sempre ordenam, incorretamente, valores `NULL` em primeiro independente da ordem escolhida.

3.3.4.7 Combinação de padrões

O MySQL fornece combinação de padrões do SQL bem como na forma de combinação de padrões baseado nas expressões regulares extendidas similares àquelas usadas pelos utilitários Unix como o `vi`, `grep` e `sed`.

A combinação de padrões SQL lhe permite você usar `_` para coincidir qualquer caractere simples e `%` para coincidir um número arbitrário de caracteres (incluindo zero caracter). No MySQL, padrões SQL são caso insensitivo por padrão. Alguns exemplos são vistos abaixo. Perceba que você não usa `=` ou `!=` quando usar padrões SQL; use os operadores de comparação `LIKE` ou `NOT LIKE` neste caso.

Para encontrar nomes começando com '`b`':

```
mysql> SELECT * FROM pet WHERE name LIKE "b%";
+--------+--------+---------+------+------------+------------+
| name   | owner  | species | sex  | birth      | death      |
+--------+--------+---------+------+------------+------------+
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+
```

Para encontrar nomes com o final '`fy`':

```
mysql> SELECT * FROM pet WHERE name LIKE "%fy";
+--------+--------+---------+------+------------+-------+
| name   | owner  | species | sex  | birth      | death |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat     | f    | 1993-02-04 | NULL  |
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

Para encontrar nomes contendo um '`w`':

```
mysql> SELECT * FROM pet WHERE name LIKE "%w%";
+----------+-------+---------+------+------------+------------+
| name     | owner | species | sex  | birth      | death      |
+----------+-------+---------+------+------------+------------+
| Claws    | Gwen  | cat     | m    | 1994-03-17 | NULL       |
| Bowser   | Diane | dog     | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL       |
+----------+-------+---------+------+------------+------------+
```

Para encontrar nomes contendo exatamente cinco caracteres, use cinco instâncias do caracter '`_`':

```
mysql> SELECT * FROM pet WHERE name LIKE "_____";
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

O outro tipo de combinação de padrões fornecido pelo MySQL usa expressões regulares extendidas. Quando você testa por uma combinação para este tipo de padrão, utilize os operadores `REGEXP` e `NOT REGEXP` (ou `RLIKE` e `NOT RLIKE`, que são sinônimos).

Algumas caracteríśticas das expressões regulares extendidas são:

- '`.`' combina qualquer caractere único

- Uma classe de caracteres '`[...]`' combina qualquer caractere que consta dentro dos colchetes. Por exemplo, '`[abc]`' combina com '`a`', '`b`', ou '`c`'. Para nomear uma sequência de caracteres utilize um traço. '`[a-z]`' combina com qualquer letra e '`[0-9]`' combina com qualquer dígito.
- '`*`' combina com nenhuma ou mais instâncias de sua precedência. Por exemplo, '`x*`' combina com qualquer número de caracteres '`x`', '`[0-9]*`' combina com qualquer número de dígitos e '`.*`' combina com qualquer número de qualquer coisa.
- Um padrão `REGEXP` casa com sucesso se ele ocorre em algum lugar no valor sendo testado. (Ele difere do padrão `LIKE`, que só obtem suceeso se eles combinarem com todo o valor.)
- Para fazer com que um padrão deva combinar com o começo ou o fim de um valor sendo testado, utilize '`^`' no começo ou '`$`' no final do padrão.

Para demonstrar como expressões regulares extendidas funcionam, as consultas com `LIKE` mostradas acima foram reescritas abaixo usando `REGEXP`.

Para encontrar nomes começando com '`b`', utilize '`^`' para combinar com o começo do nome:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^b";
+--------+--------+---------+------+------------+------------+
| name   | owner  | species | sex  | birth      | death      |
+--------+--------+---------+------+------------+------------+
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+
```

Antes da versão 3.23.4 do MySQL, `REGEXP` era caso sensitivo, e a consulta anterior não iria retornar nenhum registro. Neste caso, para combinar letras '`b`' maiúsculas e minúsculas, utilize esta consulta:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^[bB]";
```

A partir do MySQL 3.23.4, se você realmente deseja forçar uma comparação `REGEXP` com caso sensitivo, utilize a palavra-chave `BINARY` para tornar uma das strings em uma string binárias. Esta consulta irá combinar somente com '`b`'s minúsculos no começo de um nome:

```
mysql> SELECT * FROM pet WHERE name REGEXP BINARY "^b";
```

Para encontrar nomes finalizados com '`fy`', utilize '`$`' para combinar com o final do nome:

```
mysql> SELECT * FROM pet WHERE name REGEXP "fy$";
+--------+--------+---------+------+------------+-------+
| name   | owner  | species | sex  | birth      | death |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat     | f    | 1993-02-04 | NULL  |
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

Para encontrar nomes contendo um '`w`', utilize esta consulta:

```
mysql> SELECT * FROM pet WHERE name REGEXP "w";
+----------+-------+---------+------+------------+------------+
| name     | owner | species | sex  | birth      | death      |
+----------+-------+---------+------+------------+------------+
```

```
| Claws    | Gwen  | cat     | m    | 1994-03-17 | NULL       |
| Bowser   | Diane | dog     | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL       |
+----------+-------+---------+------+------------+------------+
```

Como uma expressão regular extendida encontra padrões coincidentes se eles ocorrem em qualquer lugar no valor comparado, não é necessário utiliar, na consulta anterior, nenhum metacaracter em nenhum dos lados do padrão para fazê-lo coincidir com todo o valor, como seria feito se fosse utilizado o padrão SQL.

Para encontrar nomes contendo exatamente cinco caracteres, utilize '`^`' e '`$`' para combinar com o começo e fim do nome e cinco instâncias de '`.`' entre eles.

```
mysql> SELECT * FROM pet WHERE name REGEXP "^.....$";
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

Você pode também escrever a consulta anterior utilizando o operador '`{n}`' "repete-`n`-vezes":

```
mysql> SELECT * FROM pet WHERE name REGEXP "^.{5}$";
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

3.3.4.8 Contando Registros

Bancos de dados normalmente são usados para responder a perguntas, "Qual a frequência que certo tipo de dados ocorre em uma tabela?" Por exemplo, você deve querer saber quantos animais tem, ou quantos animais cada dono tem, ou você pode querer fazer vários outros tipos de operações de censo com seus animais.

Contando o número total de animais que você tem é a mesma questão como em "Quantos registros existem na tabela `pet`?" porque existe um registro por animal. `COUNT(*)` conta o número de resultados não-`NULL`, portanto a pesquisa para contar seus animais parecerá com isto:

```
mysql> SELECT COUNT(*) FROM pet;
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
```

Logo, você recuperará os nomes das pessoas que possuam animais. Você pode usar `COUNT()` se você desejar encontrar quantos animais cada dono possui:

```
mysql> SELECT owner, COUNT(*) FROM pet GROUP BY owner;
+--------+----------+
| owner  | COUNT(*) |
+--------+----------+
| Benny  |        2 |
| Diane  |        2 |
| Gwen   |        3 |
| Harold |        2 |
+--------+----------+
```

Perceba o uso de `GROUP BY` para agrupar todos os registros para cada `owner` (dono). Sem ele, você teria uma mensagem de erro:

```
mysql> SELECT owner, COUNT(*) FROM pet;
ERROR 1140: Mixing of GROUP columns (MIN(),MAX(),COUNT()...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

`COUNT()` e `GROUP BY` são úteis para personalizar seus dados de diversas maneiras. Os seguintes exemplos mostram diferentes maneiras para realizar operações de censo nos animais.

Número de animais por espécie:

```
mysql> SELECT species, COUNT(*) FROM pet GROUP BY species;
+---------+----------+
| species | COUNT(*) |
+---------+----------+
| bird    |        2 |
| cat     |        2 |
| dog     |        3 |
| hamster |        1 |
| snake   |        1 |
+---------+----------+
```

Número de animais por sexo:

```
mysql> SELECT sex, COUNT(*) FROM pet GROUP BY sex;
+------+----------+
| sex  | COUNT(*) |
+------+----------+
| NULL |        1 |
| f    |        4 |
| m    |        4 |
+------+----------+
```

(Nesta saída, `NULL` indica que o sexo é desconhecido.)

Número de animais combinando espécie e sexo:

```
mysql> SELECT species, sex, COUNT(*) FROM pet GROUP BY species, sex;
+---------+------+----------+
| species | sex  | COUNT(*) |
+---------+------+----------+
| bird    | NULL |        1 |
| bird    | f    |        1 |
```

```
| cat     | f    |        1 |
| cat     | m    |        1 |
| dog     | f    |        1 |
| dog     | m    |        2 |
| hamster | f    |        1 |
| snake   | m    |        1 |
+---------+------+----------+
```

Não é necessário selecionar uma tabela inteira quando estiver usando `COUNT()`. Por exemplo, a consulta anterior, quando realizada apenas procurando por cachorros e gatos, se parece com isto:

```
mysql> SELECT species, sex, COUNT(*) FROM pet
    -> WHERE species = "dog" OR species = "cat"
    -> GROUP BY species, sex;
+---------+------+----------+
| species | sex  | COUNT(*) |
+---------+------+----------+
| cat     | f    |        1 |
| cat     | m    |        1 |
| dog     | f    |        1 |
| dog     | m    |        2 |
+---------+------+----------+
```

Ou se você desejar saber o número de animais por sexo somente de animais com sexo conhecido:

```
mysql> SELECT species, sex, COUNT(*) FROM pet
    -> WHERE sex IS NOT NULL
    -> GROUP BY species, sex;
+---------+------+----------+
| species | sex  | COUNT(*) |
+---------+------+----------+
| bird    | f    |        1 |
| cat     | f    |        1 |
| cat     | m    |        1 |
| dog     | f    |        1 |
| dog     | m    |        2 |
| hamster | f    |        1 |
| snake   | m    |        1 |
+---------+------+----------+
```

3.3.4.9 Utilizando Múltiplas Tabelas

A tabela `pet` mantém informações de quais animais você tem. Se você deseja gravar outras informações sobre eles como eventos em suas vidas, tais como visitas ao veterinário ou sobre suas crias, você necessitará de outra tabela. Como esta tabela deve se parecer ? Ela precisa:

- Conter o nome do animal para que você saiba a qual animal pertence o evento.
- Uma data para que você saiba quando ocorreu o evento.

- Um campo para descrever o evento.
- Um campo com o tipo de evento, se você desejar classificá-los por categoria.

Dadas estas considerações, a instrução `CREATE TABLE` para a tabela `event` deve se parecer com isto:

```
mysql> CREATE TABLE event (name VARCHAR(20), date DATE,
    -> type VARCHAR(15), remark VARCHAR(255));
```

Como na tabela `pet`, é mais fácil carregar os registros iniciais criando um arquivo texto delimitado por tabulações contendo a informação:

name	**date**	**type**	**remark**
Fluffy	1995-05-15	litter	4 kittens, 3 female, 1 male
Buffy	1993-06-23	litter	5 puppies, 2 female, 3 male
Buffy	1994-06-19	litter	3 puppies, 3 female
Chirpy	1999-03-21	vet	needed beak straightened
Slim	1997-08-03	vet	broken rib
Bowser	1991-10-12	kennel	
Fang	1991-10-12	kennel	
Fang	1998-08-28	birthday	Gave him a new chew toy
Claws	1998-03-17	birthday	Gave him a new flea collar
Whistler	1998-12-09	birthday	First birthday

Carregue os registros usando:

```
mysql> LOAD DATA LOCAL INFILE "event.txt" INTO TABLE event;
```

Baseado no que você já aprendeu com as consultas realizadas na tabela `pet`, você deve estar apto para realizar pesquisas na tabela `event`; os princípios são o mesmo. Mas quando a tabela `event`, sozinha, é insuficiente para responder às suas questões?

Suppose you want to find out the ages at which each pet had its litters. We saw earlier how to calculate ages from two dates. The litter date of the mother is in the `event` table, but to calculate her age on that date you need her birth date, which is stored in the `pet` table. This means the query requires both tables:

Suponha que você deseje descobrir as idades de cada animal quando eles tiveram cria. Nós vemos logo que é possível calcular a idade a partir das duas datas. A idade dos filhotes está na tabela `event`, mas para calcular a idade da mãe, você precisará da data de nascimento dela, que está armazenado na tabela `pet`. Isto significa que você precisará das duas tabelas para a consulta:

```
mysql> SELECT pet.name,
    -> (YEAR(date)-YEAR(birth)) - (RIGHT(date,5)<RIGHT(birth,5)) AS age,
    -> remark
    -> FROM pet, event
    -> WHERE pet.name = event.name AND type = "litter";
+--------+------+-----------------------------+
| name   | age  | remark                      |
+--------+------+-----------------------------+
| Fluffy |    2 | 4 kittens, 3 female, 1 male |
| Buffy  |    4 | 5 puppies, 2 female, 3 male |
| Buffy  |    5 | 3 puppies, 3 female         |
```

```
+--------+------+----------------------------+
```

Existem várias coisas que devem ser percebidas sobre esta consulta:

- A cláusula `FROM` lista as duas tabelas porque a consulta precisa extrair informação de ambas.
- Quando combinar (unir) informações de múltiplas tabelas, você precisa especificar como registros em uma tabela podem ser coincididas com os registros na outra. Isto é simples porque ambas possuem uma coluna `name`. A consulta utiliza a cláusula `WHERE` para coincidir registros nas duas tabelas baseadas nos valores de `name`.
- Como a coluna `name` ocorre em ambas tabelas, você deve especificar qual a tabela a que você está se referindo. Isto é feito usando o nome da tabela antes do nome da coluna separados por um ponto (`.`).

Você não precisa ter duas tabelas diferentes para realizar uma união. Algumas vezes é útil unir uma tabela a ela mesma, se você deseja comparar registros em uma tabela com outros registros na mesma tabela. Por exemplo, para encontrar pares entre seus animais, você pode unir a tabela `pet` com ela mesma para produzir pares candidatos de machos e fêmeas de acordo com as espécies:

```
mysql> SELECT p1.name, p1.sex, p2.name, p2.sex, p1.species
    -> FROM pet AS p1, pet AS p2
    -> WHERE p1.species = p2.species AND p1.sex = "f" AND p2.sex = "m";
+--------+------+--------+------+---------+
| name   | sex  | name   | sex  | species |
+--------+------+--------+------+---------+
| Fluffy | f    | Claws  | m    | cat     |
| Buffy  | f    | Fang   | m    | dog     |
| Buffy  | f    | Bowser | m    | dog     |
+--------+------+--------+------+---------+
```

Nesta consulta, nós especificamos apelidos para os nomes das tabelas para conseguir referenciar às colunas e manter com qual instância da tabela cada coluna de referência está associada.

3.4 Obtendo Informações Sobre Bancos de Dados e Tabelas

E se você esquecer o nome de um banco de dados ou tabela, ou como é a estrutura de uma certa tabela (por exemplo, como suas colunas são chamadas)? O MySQL resolve este problema através de diversas instruções que fornecem informações sobre os bancos de dados e as tabelas que ele suporta.

Você já viu `SHOW DATABASES`, que lista os bancos de dados gerenciados pelo servidor. Para saber qual banco de dados está sendo usado atualmente, utilize a função `DATABASE()`:

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+
```

Se você ainda não selecionou nenhum banco de dados ainda, o resultado é `NULL`. (ou a string vazia antes do MySQL 4.1.1).

Para saber quais tabelas o banco de dados atual contêm (por exemplo, quando você não tem certeza sobre o nome de uma tabela), utilize este comando:

```
mysql> SHOW TABLES;
+----------------------+
| Tables in menagerie |
+----------------------+
| event                |
| pet                  |
+----------------------+
```

Se você deseja saber sobre a estrutura de uma tabela, o comando `DESCRIBE` é útil; ele mostra informações sobre cada uma das colunas da tabela:

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

A coluna `Field` (campo) indica o nome da coluna, `Type` é o tipo de dados para a coluna, `Null` indica se a coluna pode conter valores nulos (`NULL`), `key` indica se a coluna é indexada ou não e `Default` especifica o valor padrão da coluna.

Se você tem índices em uma tabela, `SHOW INDEX FROM tbl_nome` traz informações sobre eles.

3.5 Utilizando `mysql` em Modo Batch

Nas seções anteriores, você usou `mysql` interativamente para fazer consultas e ver os resultados. Você pode também executar `mysql` no modo batch. Para fazer isto, coloque os comando que você deseja executar em um arquivo, e diga ao `mysqld` para ler sua entrada do arquivo:

```
shell> mysql < batch-file
```

Se você estiver executando o `mysql` no Windows e tiver algum caracter especial no arquivo que provocou o problema, você pode fazer:

```
dos> mysql -e "source batch-file"
```

Se você precisa especificar parâmetros de conexão na linha de comando, o comando deve parecer com isto:

```
shell> mysql -h host -u user -p < batch-file
Enter password: ********
```

Quando você utilizar o `mysql` desta forma, você estará criando um arquivo script, depois executando o script.

Se você quiser que o script continue mesmo se hopuver erros, você deve usar a opção de linha de comando `--force`.

Por que usar um script? Existem várias razões:

- Se você executa uma query repetidamente (digamos, todos os dias ou todas as semanas), transformá-lo em um script permite que você não o redigite toda vez que o executa.
- Você pode gerar novas consultas a partir das já existentes copiando e editando os arquivos de script.
- O modo batch pode também ser útil quando você estiver desenvolvendo uma consulta, particularmente para comandos de múltiplas linhas ou sequências de comandos com várias instruções. Se você cometer um erro, não será necessário redigitar tudo. Apenas edite seu arquivo script e corrija o erro, depois diga ao `mysql` para executá-lo novamente.
- Se você tem uma query que produz muita saída, você pode encaminhar a saída através de um páginador.

  ```
  shell> mysql < batch-file | more
  ```

- Você pode capturar a saída em um arquivo para processamento posterior:

  ```
  shell> mysql < batch-file > mysql.out
  ```

- Você pode distribuir seu script para outras pessoas para que elas possam executar os comandos também.
- Algumas situações não permitem uso interativo, por exemplo, quando você executa uma consulta através de um processo automático (`cron` job). Neste caso, você deve usar o modo batch.

A formato padrão de saída é diferente (mais conciso) quando você executa o `mysql` no modo batch do que quando você o usa interativamente. Por exemplo, a saída de `SELECT DISTINCT species FROM pet` se parece com isto quando você o executa interativamente:

```
+---------+
| species |
+---------+
| bird    |
| cat     |
| dog     |
| hamster |
| snake   |
+---------+
```

Mas fica assim quando você o executa no modo batch:

```
species
bird
cat
dog
hamster
snake
```

Se você desejar obter o formato de saída interativa no modo batch, utilize `mysql -t`. Para mostrar a saída dos comandos que são executados, utilize `mysql -vvv`.

Você também pode utilizar scripts no prompt de linha de comando `mysql` usando o comando `source`:

```
mysql> source filename;
```

3.6 Exemplos de Consultas Comuns

Aqui estão os exemplos de como resolver problemas comuns com o MySQL.

Alguns dos exemplos usam a tabela `shop` para armazenar o preço de cada ítem (article) para certas revendas (dealers). Supondo que cada revenda tenha um preço fixo por artigo, então (`article`, `dealer`) é uma chave primária para os registros.

Inicie a ferramenta de linha de comando `mysql` e selecione um banco de dados:

```
shell> mysql o-nome-do-seu-banco-de-dados
```

(Na maioria das instalações do MySQL, você pode usar o banco de dados `test`).

Você pode criar e popular a tabela exemplo assim:

```
mysql> CREATE TABLE shop (
    -> article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
    -> dealer  CHAR(20)                 DEFAULT ''     NOT NULL,
    -> price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL,
    -> PRIMARY KEY(article, dealer));
mysql> INSERT INTO shop VALUES
    -> (1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),(3,'C',1.69),
    -> (3,'D',1.25),(4,'D',19.95);
```

Depois de executar as instruções a tabela deve ter o seguinte conteúdo:

```
mysql> SELECT * FROM shop;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | A      |  3.45 |
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | B      |  1.45 |
|    0003 | C      |  1.69 |
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

3.6.1 O Valor Máximo para uma Coluna

"Qual é o maior número dos ítens?"

```
SELECT MAX(article) AS article FROM shop;

+---------+
| article |
```

```
+---------+
|       4 |
+---------+
```

3.6.2 O Registro que Armazena o Valor Máximo para uma Coluna Determinada

"Encontre o número, fornecedor e preço do ítem mais caro."

No SQL ANSI isto é feito fácilmente com uma sub-consulta:

```
SELECT article, dealer, price
FROM   shop
WHERE  price=(SELECT MAX(price) FROM shop);
```

No MySQL (que ainda não suporta sub-selects), faça isto em dois passos:

1. Obtenha o valor do preço máximo da tabela com uma instrução `SELECT`.

```
mysql> SELECT MAX(price) FROM shop;
+------------+
| MAX(price) |
+------------+
|      19.95 |
+------------+
```

2. Usando o valor 19.95 mostrado pela consulta anterior como o preço máximo do artigo, grave uma consulta para localizar e mostrar o registro correspondente:

```
mysql> SELECT article, dealer, price
    -> FROM   shop
    -> WHERE  price=19.95;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

Outra solução é ordenar todos os registros por preço de forma descendente e obtenha somente o primeiro registro utilizando a cláusula específica do MySQL `LIMIT`:

```
SELECT article, dealer, price
FROM   shop
ORDER BY price DESC
LIMIT 1;
```

NOTA: Se existir diversos ítens mais caros, cada um com um preço de 19.95, a solução `LIMIT` mostra somente um deles !

3.6.3 Máximo da Coluna por Grupo

"Qual é o maior preço por ítem?"

```
SELECT article, MAX(price) AS price
FROM   shop
GROUP BY article

+---------+-------+
| article | price |
+---------+-------+
|    0001 |  3.99 |
|    0002 | 10.99 |
|    0003 |  1.69 |
|    0004 | 19.95 |
+---------+-------+
```

3.6.4 As Linhas Armazenando o Group-wise Máximo de um Certo Campo

"Para cada ítem, encontre o(s) fornecedor(s) com o maior preço."

No SQL-99 (e MySQL 4.1 ou superior), o problema pode ser solucionado com uma subconsulta como esta:

```
SELECT article, dealer, price
FROM   shop s1
WHERE  price=(SELECT MAX(s2.price)
              FROM shop s2
              WHERE s1.article = s2.article);
```

Em versões anteriores a do MySQL 4.1 é melhor fazê-lo em diversos passos:

1. Obtenha a lista de pares (article,maxprice).
2. Para cada ítem, obtenha os registros correspondentes que tenham o maior preço.

Isto pode ser feito facilmente com uma tabela temporária e um join:

```
CREATE TEMPORARY TABLE tmp (
        article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
        price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL);

LOCK TABLES shop READ;

INSERT INTO tmp SELECT article, MAX(price) FROM shop GROUP BY article;

SELECT shop.article, dealer, shop.price FROM shop, tmp
WHERE shop.article=tmp.article AND shop.price=tmp.price;

UNLOCK TABLES;

DROP TABLE tmp;
```

Se você não usar uma tabela `TEMPORÁRIA`, você deve bloquear também a tabela `tmp`.

"Posso fazer isto com uma única query?"

Sim, mas somente com um truque ineficiente chamado "truque MAX-CONCAT":

```
SELECT article,
       SUBSTRING( MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 7) AS dealer,
  0.00+LEFT(      MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 6) AS price
FROM   shop
GROUP BY article;

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | C      |  1.69 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

O último exemplo pode, é claro, ser feito de uma maneira mais eficiente fazendo a separação da coluna concatenada no cliente.

3.6.5 Utilizando Variáveis de Usuário

Você pode usar variáveis de usuários no MySQL para lembrar de resultados sem a necessidade de armazená-las em variáveis no cliente. Veja Seção 6.1.4 [Variables], Página 474.

Por exemplo, para encontrar os ítens com os preços mais altos e mais baixos você pode fazer isto:

```
select @min_price:=min(price),@max_price:=max(price) from shop;
select * from shop where price=@min_price or price=@max_price;

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

3.6.6 Utilizando Chaves Estrangeiras

No MySQL 3.23.44 e acima, tabelas `InnoDB` suportam verificação de restrições de chaves estrangerias. Veja Seção 7.5 [InnoDB], Página 643. Veja também Seção 1.8.4.5 [ANSI diff Foreign Keys], Página 50.

Você não precisa de chaves estrangeiras para unir 2 tabelas. Para outros tipos de tabela diferentes de `InnoDB`, As únicas coisas que o MySQL atualmente não faz são 1) `CHECK`, para ter certeza que as chaves que você usa realmente existem na tabela ou tabelas referenciadas e 2) apagar automaticamente registros da tabela com uma definição de chave estrangeira. Usando suas chaves para unir a tabela funcionará bem:

```
CREATE TABLE person (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    colour  ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL REFERENCES person(id),
    PRIMARY KEY (id)
);


INSERT INTO person VALUES (NULL, 'Antonio Paz');

INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', LAST_INSERT_ID()),
(NULL, 'dress', 'white', LAST_INSERT_ID()),
(NULL, 't-shirt', 'blue', LAST_INSERT_ID());


INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');

INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', LAST_INSERT_ID()),
(NULL, 'polo', 'red', LAST_INSERT_ID()),
(NULL, 'dress', 'blue', LAST_INSERT_ID()),
(NULL, 't-shirt', 'white', LAST_INSERT_ID());


SELECT * FROM person;
+----+---------------------+
| id | name                |
+----+---------------------+
|  1 | Antonio Paz         |
|  2 | Lilliana Angelovska |
+----+---------------------+

SELECT * FROM shirt;
+----+---------+--------+-------+
| id | style   | colour | owner |
+----+---------+--------+-------+
|  1 | polo    | blue   |     1 |
|  2 | dress   | white  |     1 |
|  3 | t-shirt | blue   |     1 |
|  4 | dress   | orange |     2 |
|  5 | polo    | red    |     2 |
```

```
|  6 | dress   | blue   |     2 |
|  7 | t-shirt | white  |     2 |
+----+---------+--------+-------+


SELECT s.* FROM person p, shirt s
 WHERE p.name LIKE 'Lilliana%'
   AND s.owner = p.id
   AND s.colour  <> 'white';

+----+-------+--------+-------+
| id | style | colour | owner |
+----+-------+--------+-------+
|  4 | dress | orange |     2 |
|  5 | polo  | red    |     2 |
|  6 | dress | blue   |     2 |
+----+-------+--------+-------+
```

3.6.7 Pesquisando em Duas Chaves

O MySQL ainda não otimiza quando você pesquisa em duas chaves diferentes combinadas com `OR` (Pesquisa em uma chave com diferentes partes `OR` é muito bem otimizadas).

```
SELECT field1_index, field2_index FROM test_table
WHERE field1_index = '1' OR  field2_index = '1'
```

A razão é que nós ainda não tivemos tempos para fazer este tratamento de uma maneira eficiente no caso geral. (A manipulação do `AND` é, em comparação, completamente geral e funciona muito bem).

No MySQL 4.0 e acimo, você pode solucionar este problema eficientemente usando um `UNION` que combina a saída de duas instruções `SELECT` separadas. Veja Seção 6.4.1.2 [UNION], Página 569. Cada `SELECT` busca apenas uma chave e pode ser otimizada.

```
SELECT field1_index, field2_index FROM test_table WHERE field1_index = '1'
UNION
SELECT field1_index, field2_index FROM test_table WHERE field2_index = '1';
```

Em versões do MySQL anteirores a 4.0, você pode conseguir o mesmo efeito usando uma tabela `TEMPORARY` e instruções `SELECT` separadas. Este tipo de otimização também é muito boa se você estiver utilizando consultas muito complicadas no qual o servidor SQL faz as otimizações na ordem errada.

```
CREATE TEMPORARY TABLE tmp
SELECT field1_index, field2_index FROM test_table WHERE field1_index = '1';
INSERT INTO tmp
SELECT field1_index, field2_index FROM test_table WHERE field2_index = '1';
SELECT * from tmp;
DROP TABLE tmp;
```

A maneira descrita acima para resolver esta consulta é uma união (`UNION`) de duas consultas.

3.6.8 Calculando Visitas Diárias

O seguinte exemplo mostra como você pode usar as funções binárias de agrupamento para calcular o número de dias por mês que um usuário tem visitado uma página web.

```
CREATE TABLE t1 (year YEAR(4), month INT(2) UNSIGNED ZEROFILL, day INT(2) UNSIGNED
INSERT INTO t1 VALUES(2000,1,1),(2000,1,20),(2000,1,30),(2000,2,2),(2000,2,23),(200
```

A tabela exemplo contém valores ano-mês-dia representando visitas feitas pelos usuários a página. Para determinar quantos quantos dias diferentes em cada mês estas visitas ocorriam, use esta consulta:

```
SELECT year,month,BIT_COUNT(BIT_OR(1<<day)) AS days FROM t1 GROUP BY year,month;
```

que retornará:

```
+------+-------+------+
| year | month | days |
+------+-------+------+
| 2000 |    01 |    3 |
| 2000 |    02 |    2 |
+------+-------+------+
```

O exemplo acima calcula quantos dias diferentes foram usados para uma combinação fornecida de mês/ano, com remoção automática de entradas duplicadas.

3.6.9 Usando AUTO_INCREMENT

O atributo AUTO_INCREMENT pode ser usado para gerar uma identificação única para um novo registro:

```
CREATE TABLE animals (
             id MEDIUMINT NOT NULL AUTO_INCREMENT,
             name CHAR(30) NOT NULL,
             PRIMARY KEY (id)
             );
INSERT INTO animals (name) VALUES ("dog"),("cat"),("penguin"),
                                  ("lax"),("whale"),("ostrich");
SELECT * FROM animals;
```

Que retorna:

```
+----+---------+
| id | name    |
+----+---------+
|  1 | dog     |
|  2 | cat     |
|  3 | penguin |
|  4 | lax     |
|  5 | whale   |
|  6 | ostrich |
+----+---------+
```

Você pode recuperar o valor AUTO_INCREMENT mais recente com a função SQL LAST_INSERT_ID() ou a função da API C mysql_insert_id(). Nota: para uma inserção de várias

linhas `LAST_INSERT_ID()`/`mysql_insert_id()` retornará atualmente a `AUTO_INCREMENT` chave da **primeira** linha inserida. Isto permite que inserções multi-linhas sejam reproduzidas corretamente em outros servidores em uma configração de replicação.

Para tabelas `MyISAM` e `BDB` você pode especificar `AUTO_INCREMENT` em uma coluna secundária em um índice multi-coluna. Neste caso, o valor gerado para a coluna `AUTO_INCREMENT` é calculado como `MAX(auto_increment_column)+1) WHERE prefix=given-prefix`. Isto é útil quando você quer colocar dados em grupos ordenados.

```
CREATE TABLE animals (
             grp ENUM('fish','mammal','bird') NOT NULL,
             id MEDIUMINT NOT NULL AUTO_INCREMENT,
             name CHAR(30) NOT NULL,
             PRIMARY KEY (grp,id)
             );
INSERT INTO animals (grp,name) VALUES("mammal","dog"),("mammal","cat"),
                   ("bird","penguin"),("fish","lax"),("mammal","whale"),
                     ("bird","ostrich");

SELECT * FROM animals ORDER BY grp,id;
```

Que retorna:

```
+--------+----+---------+
| grp    | id | name    |
+--------+----+---------+
| fish   |  1 | lax     |
| mammal |  1 | dog     |
| mammal |  2 | cat     |
| mammal |  3 | whale   |
| bird   |  1 | penguin |
| bird   |  2 | ostrich |
+--------+----+---------+
```

Note que neste caso (quando o valor `AUTO_INCREMENT` é parte de um índice multi-coluna), o valor de `AUTO_INCREMENT` será reutilizado se você deletar a linha com o maior valor `AUTO_INCREMENT` em qualquer grupo. Isto caontece mesmo para tabelas `MyISAM`, para as quais os valores `AUTO_INCREMENT` normalmente não são reusados.)

3.7 Consultas de Projetos Gêmeos

Em Analytikerna e Lentus, nós estamos fazendo os sistemas e trabalho de campo para um grande projeto de pesquisa. Este projeto é uma colaboração entre o Institudo de Medicina Ambiental em Karolinksa Institutet Stockholm e a Seção de Pesquisa Clínica em Envelhecimento e Psicologia na University of Southern California.

O projeto envolve uma parte de seleção onde todos os gêmeos na Suécia mais velhos que 65 anos são entrevistados por telefone. Gêmeos que preenchem certos critérios passam para o próximo estágio. Neste estágio posterior, gêmeos que desejam participar são visitados por uma equipe de doutores/enfermeiros. Alguns dos consultas incluem exames físicos e neuropsicológico, testes de laboratório, imagem neural, determinação do estado psicológico

e coletas de histórico familiar. Adicionalmente, dados são coletados em fatores de riscos médicos e ambientais.

Mais informações sobre o estudos dos gêmeos pode ser encontrados em: `http://www.mep.ki.se/twinreg/index_en.html`

A parte posterior do projeto é administrada com uma interface Web escrita utilizando a linguagem Perl e o MySQL.

Cada noite todos dados das entrevistas são movidos para um banco de dados MySQL.

3.7.1 Encontrando Todos Gêmeos Não-distribuídos

A seguinte consulta é usada para determinar quem vai na segunda parte do projeto:

```
SELECT
        CONCAT(p1.id, p1.tvab) + 0 AS tvid,
        CONCAT(p1.christian_name, " ", p1.surname) AS Name,
        p1.postal_code AS Code,
        p1.city AS City,
        pg.abrev AS Area,
        IF(td.participation = "Aborted", "A", " ") AS A,
        p1.dead AS dead1,
        l.event AS event1,
        td.suspect AS tsuspect1,
        id.suspect AS isuspect1,
        td.severe AS tsevere1,
        id.severe AS isevere1,
        p2.dead AS dead2,
        l2.event AS event2,
        h2.nurse AS nurse2,
        h2.doctor AS doctor2,
        td2.suspect AS tsuspect2,
        id2.suspect AS isuspect2,
        td2.severe AS tsevere2,
        id2.severe AS isevere2,
        l.finish_date
FROM
        twin_project AS tp
        /* For Twin 1 */
        LEFT JOIN twin_data AS td ON tp.id = td.id
                  AND tp.tvab = td.tvab
        LEFT JOIN informant_data AS id ON tp.id = id.id
                  AND tp.tvab = id.tvab
        LEFT JOIN harmony AS h ON tp.id = h.id
                  AND tp.tvab = h.tvab
        LEFT JOIN lentus AS l ON tp.id = l.id
                  AND tp.tvab = l.tvab
        /* For Twin 2 */
        LEFT JOIN twin_data AS td2 ON p2.id = td2.id
```

```
          AND p2.tvab = td2.tvab
        LEFT JOIN informant_data AS id2 ON p2.id = id2.id
          AND p2.tvab = id2.tvab
        LEFT JOIN harmony AS h2 ON p2.id = h2.id
          AND p2.tvab = h2.tvab
        LEFT JOIN lentus AS l2 ON p2.id = l2.id
          AND p2.tvab = l2.tvab,
        person_data AS p1,
        person_data AS p2,
        postal_groups AS pg
WHERE
        /* p1 gets main twin and p2 gets his/her twin. */
        /* ptvab is a field inverted from tvab */
        p1.id = tp.id AND p1.tvab = tp.tvab AND
        p2.id = p1.id AND p2.ptvab = p1.tvab AND
        /* Just the sceening survey */
        tp.survey_no = 5 AND
        /* Skip if partner died before 65 but allow emigration (dead=9) */
        (p2.dead = 0 OR p2.dead = 9 OR
         (p2.dead = 1 AND
          (p2.death_date = 0 OR
           (((TO_DAYS(p2.death_date) - TO_DAYS(p2.birthday)) / 365)
            >= 65))))
        AND
        (
        /* Twin is suspect */
        (td.future_contact = 'Yes' AND td.suspect = 2) OR
        /* Twin is suspect - Informant is Blessed */
        (td.future_contact = 'Yes' AND td.suspect = 1
                                   AND id.suspect = 1) OR
        /* No twin - Informant is Blessed */
        (ISNULL(td.suspect) AND id.suspect = 1
                            AND id.future_contact = 'Yes') OR
        /* Twin broken off - Informant is Blessed */
        (td.participation = 'Aborted'
         AND id.suspect = 1 AND id.future_contact = 'Yes') OR
        /* Twin broken off - No inform - Have partner */
        (td.participation = 'Aborted' AND ISNULL(id.suspect)
                                      AND p2.dead = 0))
        AND
        l.event = 'Finished'
        /* Get at area code */
        AND SUBSTRING(p1.postal_code, 1, 2) = pg.code
        /* Not already distributed */
        AND (h.nurse IS NULL OR h.nurse=00 OR h.doctor=00)
        /* Has not refused or been aborted */
        AND NOT (h.status = 'Refused' OR h.status = 'Aborted'
```

```
            OR h.status = 'Died' OR h.status = 'Other')
    ORDER BY
            tvid;
```

Algumas explicações:

`CONCAT(p1.id, p1.tvab) + 0 AS tvid`
: N queremos ordenar o `id` e o `tvab` concatenados na ordem numérica. Adicionando `0` ao resultado faz o MySQL tratar o resultado como um número.

coluna `id`
: Esta identifica um par de gêmeos. Ela é uma chave em todas as tabelas.

column `tvab`
: Esta identifica um gêmeo em um par. Ela pode ter um valor de `1` ou `2`.

column `ptvab`
: Esta é o inverso de `tvab`. Quando `tvab` é `1` este campo é `2` e vice versa. Ela existe para poupar digitação e tornar mais fácil para o MySQL otimizar a query.

Esta consulta demonstra, entre outras coisas, como fazer buscas em uma tabela a partir da mesma tabela com uma uniao (`p1` e `p2`). No exemplo, isto é usado para conferir se um par de um gêmeo morreu antes de 65 anos. Se for verdade, a linha não é retornada.

Tudo acima existe em todas as tabelas com informações relacionada aos gêmeos. Nós temos uma chave em ambos `id,tvab` (todas as tabelas) e `id,ptvab` (`person_data`) para tornar as consultas mais rápidas.

Na nossa máquina de produção (Um UltraSPARC 200MHz), esta consulta retorna entre 150-200 linhas e gasta menos que um segundo.

O número atual de registros nas tabelas usadas acima:

Tabela	**Registros**
`person_data`	71074
`lentus`	5291
`twin_project`	5286
`twin_data`	2012
`informant_data`	663
`harmony`	381
`postal_groups`	100

3.7.2 Mostrando uma Tabela sobre a Situação dos Pares Gêmeos

Cada entrevista termina com um código da situação chamado `event`. A consulta mostrada abaixa é usada para mostrar uma tabela sobre todos pares gêmeos combinados por evento. Ela indica em quantos pares ambos gêmeos terminaram, em quantos pares um gêmeo terminou e o outro foi recusado e assim por diante.

```
    SELECT
            t1.event,
            t2.event,
            COUNT(*)
    FROM
            lentus AS t1,
```

```
        lentus AS t2,
        twin_project AS tp
WHERE
        /* We are looking at one pair at a time */
        t1.id = tp.id
        AND t1.tvab=tp.tvab
        AND t1.id = t2.id
        /* Just the sceening survey */
        AND tp.survey_no = 5
        /* This makes each pair only appear once */
        AND t1.tvab='1' AND t2.tvab='2'
GROUP BY
        t1.event, t2.event;
```

3.8 Utilizando MySQL com Apache

Existem programas que lhe permite autenticar seus usuários a partir de um banco de dados MySQL e também permite gravar seus arquivos de log em uma tabela MySQL.

Você pode alterar o formato de log do Apache para ser facilmente lido pelo MySQL colocando o seguinte no arquivo de configuração do Apache:

```
LogFormat \
        "\"%h\",%{%Y%m%d%H%M%S}t,%>s,\"%b\",\"%{Content-Type}o\",  \
        \"%U\",\"%{Referer}i\",\"%{User-Agent}i\""
```

Para carregar uma arquivo de log naquele formato dentro do MySQL, você pode usar uma instrução deste tipo:

```
LOAD DATA INFILE '/local/access_log' INTO TABLE nome_tabela
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '\\'
```

A tabela chamada deve ser criada para ter colunas que correpondem a aquelas que a linha `LogFormat` gravam no arquivo de log.

4 Administração do Bancos de Dados MySQL

4.1 Configurando o MySQL

4.1.1 Opções de Linha de Comando do `mysqld`

Na maioria dos casos você deve gerenciar as opções do mysqld por meio dos arquivos de opções. Veja Seção 4.1.2 [Arquivos de opções], Página 216.

`mysqld` e `mysqld.server` lêem opções dos grupos `mysqld` e `server`. `mysqld_safe` lê as opções dos grupos `mysqld`, `server`, `mysqld_safe` e `mysqld_safe`. Um servidor MySQL embutido normalmente lê opções do grupos `server`, `embedded` e `xxxxx_SERVER`, onde `xxxxx` é o nome da aplicação.

`mysqld` aceita os seguintes opções de linha de comando. Aqui está uma lista das mais comuns. Para uma lista completa execute `mysqld --help`. As opções usadas para replicação estào listadas em uma seção separada, veja Seção 4.11.6 [Replication Options], Página 392.

`--ansi` Utilizar a sintaxe ANSI SQL no lugar da sintaxe MySQL Veja Seção 1.8.2 [ANSI mode], Página 42.

`-b, --basedir=path`
Encaminho para o diretório de instalação. Todos os caminhos normalmente são resolvidos em relação a este.

`--big-tables`
Permite grandes conjuntos de resultados salvando todos os conjuntos temporários em um arquivo. Ele resolve a maioria dos erros 'table full', mas também abaixa a velocidade das consultas nas quais as tabelas em memória seriam suficientes. Desde a Versão 3.23.2, o MySQL é capaz de resolver isto automaticamente usando memória para pequenas tabelas temporárias e trocando para o disco as tabelas, quando for necessário.

`--bind-address=IP`
Endereço IP para ligar.

`--console`
Grava a mensagem de erro no stderr/stdout mesmo se `--log-error` é especificado. No Windows o mysqld não fechará a tela de console se esta opção é usada.

`--character-sets-dir=path`
Diretório onde estão os conjuntos de caracteres. Veja Seção 4.7.1 [Conjunto de caracteres], Página 325.

`--chroot=path`
Coloca o daemon `mysqld` no diretorio chroot durante a inicialização. Medida de segurança recomendada desde o MySQL 4.0 (MySQL 3.23 não está apto a fornecer um chroot 100% fechado. Limita os comandos `LOAD DATA INFILE` e `SELECT ... INTO OUTFILE`.

`--core-file`
: Grava um arquivo core se o `mysqld` morrer. Para alguns sistemas você deve também especificar `--core-file-size` para `mysqld_safe`. Veja Seção 4.8.2 [`mysqld_safe`], Página 331. Note que em alguns sistemas, como Solaris, você não consiguirá um arquivo core se você também estiver usando a opção `--user`.

`-h, --datadir=caminho`
: Encaminha para o diretório raiz dos bancos de dados.

`--debug[...]=`
: Se o MySQL está configurado com `--with-debug`, você pode usar esta opção para obter um arquivo de rastreamento indicando o que o `mysqld` está fazendo. Veja Seção E.1.2 [Criando arquivos trace], Página 1080.

`--default-character-set=conjunto_caracter`
: Configura o conjunto de caracteres padrão. Veja Seção 4.7.1 [Conjunto de caracteres], Página 325.

`--default-table-type=tipo`
: Configura o tipo de tabela padrão. Veja Cap“ptexi tulo 7 [Tipos de tabelas], Página 630.

`--delay-key-write[= OFF | ON | ALL]`
: Como o `DELAYED KEYS` do MyISAM deve ser usado. Veja Seção 5.5.2 [Parâmetros do servidor], Página 454.

`--delay-key-write-for-all-tables; No MySQL 4.0.3 você deve usar --delay-key-write=ALL.`
: Não descarrega buffers das chaves entre escritas em nenhuma tabela `MyISAM`. Veja Seção 5.5.2 [Parâmetros do servidor], Página 454.

`--des-key-file=filename`
: Read the default keys used by `DES_ENCRYPT()` and `DES_DECRYPT()` from this file.

`--enable-external-locking (era --enable-locking)`
: Habilita o bloqueio do sistema. Perceba que se usar esta opção em um sistema que não possui um lockd() completamente funcional (como no Linux) você pode fazer com que o mysqld entre em deadlock.

`--enable-named-pipe`
: Habilita suporte para named pipes (somente no NT/Win2000/XP).

`-T, --exit-info`
: Esta é uma máscara binária com diferêntes parâmetros que pode ser usada para depurar o servidor mysqld; Esta opção não deve ser usada por alguém que não a conheça muito bem!

`--flush` Atualiza todas as alterações no disco depois de cada comando SQL. Normalmente o MySQL só faz a escrita de todas as alterações no disco depois de cada comando SQL e deixa o sistema operacional lidar com a sincronização com o disco. Veja Seção A.4.1 [Falhas], Página 925.

`-?, --help`
Mostra uma pequena ajuda e sai.

`--init-file=arquivo`
Lê comandos SQL do arquivo especificado na inicialização.

`-L, --language=...`
Mensagens de erro do cliente na língua especificada. Pode ser fornecido como um caminho completo. Veja Seção 4.7.2 [Línguas], Página 327.

`-l, --log[=arquivo]`
Log de conexões e consultas ao arquivo. Veja Seção 4.10.2 [Log de consultas], Página 373.

`--log-bin=[arquivo]`
Registra todas as consultas que alteram dados em arquivo. Usado para backup e replicação. Veja Seção 4.10.4 [Binary log], Página 374.

`--log-bin-index[=arquivo]`
Arquivo de índice para nomes de arquivos de log binario. Veja Seção 4.10.4 [Log binário], Página 374.

`--log-error[=arquivo]`
Registra mensagens de erro e inicialização neste arquivo. Veja Seção 4.10.1 [Log de erro], Página 372.

`--log-isam[=arquivo]`
Log de todas alterações ISAM/MyISAM no arquivo (usado somente quando estiver depurando bancos ISAM/MyISAM).

`--log-long-format`
Registra algumas informações extras nos aruivos de log (log de atualizações, log binário de atualizações e log de consultas lentas, independente de qual está ativado). Por exemplo, nome do usuário e timestamp são registrados para a consulta. Se você estiver usando `--log-slow-queries` e `--log-long-format`, então consultas que não estão usando índices são registradas ao log de consultas lentas. Note que `--log-long-format` está obsoleto a partir do MySQL versão 4.1, quando `--log-short-format` foi introduzido (`--log-long-format` é a configuração padrão desde a versão 4.1). Note também que a partir do MySQL 4.1, a opção `--log-queries-not-using-indexes` está disponível para propósito de registro de consultas que não usam índices para o log de consultas lentas.

`--log-queries-not-using-indexes`
Se você estiver usando `--log-slow-queries`, então consultas que não estão usando índices estão registradas no log de consultas lentas. Esta opções está disponível a partir do MySQL 4.1. Veja Seção 4.10.5 [Slow query log], Página 377.

`--log-short-format`
Registra menos informações extras nos aruivos de log (log de atualizações, log binário de atualizações e log de consultas lentas, independente de qual está

ativado). Por exemplo, nome do usuário e timestamp são registrados para a consulta. Esta opção foi introduzida no MySQL 4.1.

`--log-slow-queries[=arquivo]`
: Log de todas as consultas que levam mais de `long_query_time` segundos de execução para um arquivo. Note que o padrão para a quantidade de informação registrada alterou no MySQL 4.1. Veja as opções `--log-long-format` e `--log-long-format` para mais detalhes. Veja Seção 4.10.5 [Slow query log], Página 377.

`--log-update[=arquivo]`
: Log de atualizações para `file.#` onde `#` é um número único se não for fornecido. Veja Seção 4.10.3 [Log de atualização], Página 373. O log de atualização estáobsoleto e será removido no MySQL 5.0; você deve usar o log binário em seu lugar (`--log-bin`). Veja Seção 4.10.4 [Log binário], Página 374. A partir da versão 5.0, usar `--log-update` apenar ligará o log binário.

`--low-priority-updates`
: Operações de alterações das tabelas (`INSERT/DELETE/UPDATE`) irão ter prioridade menor do que as selects. Isto também pode ser feito usando `{INSERT | REPLACE | UPDATE | DELETE} LOW_PRIORITY ...` para baixar a prioridade de somente uma consulta, ou `SET OPTION SQL_LOW_PRIORITY_UPDATES=1` para alterar a prioridade em uma única thread. Veja Seção 5.3.2 [Bloqueio de tabelas], Página 444.

`--memlock`
: Bloqueia o processo `mysqld` na memória. Isto funciona somente se o seu sistema suportar a chamada de sistema `mklockall()` (como no Solaris). Isto pode ajudar se você tiver um problema no qual o sistema operacional faz com que o `mysqld` faça a troca em disco. Note que o uso desta opção exige que você execute o servidor como `root`, que normalmente não é uma boa idéia por razões de segurança.

`--myisam-recover [=opção[,opção...]]] onde opção é qualquer combinação`
: de `DEFAULT`, `BACKUP`, `FORCE` ou `QUICK`. Você também pode configurar isto explicitamente para `""` se você deseja desabilitar esta opção. Se esta opção for usada, o `mysqld` irá conferir na abertura se a tabela está marcada como quebrada ou se a tabela não foi fechada corretamente. (A última opção funciona somente se você estiver executando com `--skip-locking`). Se este for o caso `mysqld` irá executar uma conferência na tabela. Se a tabela estiver corrompida, o `mysqld` irá tentar repará-la.

  As seguintes opções afetam no funcionamento da reparação.

Opção	Descrição
DEFAULT	O mesmo que não fornecer uma opção para `--myisam-recover`.
BACKUP	Se os dados da tabela foram alterados durante a recuperação, salve um backup do arquivo de dados '`nome_tabela.MYD`' como '`nome_tabela_dia_hora.BAK`'.
FORCE	Execute a recuperação mesmo se perdermos mais de uma linha do arquivo .MYD.

QUICK Não confira as linhas na tabela se não existir nenhum bloco apagado.

Antes da tabela ser reparada automaticamente, o MySQL irá adicionar uma nota no log de erros. Se você desejar que a recuperação da maioria dos problemas não tenha a intervenção de algum usuário, devem ser usadas as opções `BACKUP,FORCE`. Isto irá forçar um reparo de uma tabela mesmo se alguns registros forem apagados, mas ele manterá o arquivo de dados antigo como um backup para que você possa examinar posteriormente o que aconteceu.

`--new` A partir da versão 4.0.12, a opção `--new` pode ser usada para fazer o servidor se comportar como 4.1 em certos aspectos, facilitando a atualização da versão 4.0 para 4.1:

- `TIMESTAMP` é retornado com uma string com o formato `'YYYY-MM-DD HH:MM:SSS`. Veja Seção 6.2 [Tipos de colunas], Página 482.

`--pid-file=caminho`
Encaminha para o arquivo pid usado pelo `mysqld_safe`.

`-P, --port=...`
Número da porta para conexões TCP/IP.

`-o, --old-protocol`
Utilize o protocolo 3.20 para compatibilidade com alguns clientes muito antigos. Veja Seção 2.5.5 [Atualizando a versão 3.20], Página 129.

`--one-thread`
Usa somente uma thread (para depuração sobre Linux). Esta opção está disponível apenas se o servidor está construído com a depuração habilitada. Veja Seção E.1 [Depurando o servidor], Página 1079.

`--open-files-limit=`
Para alterar o número de descritores de arquivos disponíveis para o `mysqld`. Se isto não estiver configurado com 0, então o `mysqld` usará este valor para reservar descritores de arquivos para usar com `setrlimit()`. Se este valor é 0 então o `mysqld` reservará `max_connections*5` ou `max_connections + table_cache*2` (que é sempre é maior) número de arquivos. Você deve tentar aumentar isto se o `mysqld` lhe retornar o erro 'Too many open files'.

`-O, --set-variable=name=value`
Fornece um valor para uma variável. `--help` lista as variáveis. Você pode encontrar uma descrição completa para todas as variáveis na seção `SHOW VARIABLES` deste manual. Veja Seção 4.6.8.4 [SHOW VARIABLES], Página 309. A seção de sintonia dos parâmetros do servidor inclui informações sobre como otimizá-los. Por favor, note que `--set-variable=name=value` e `-O name=value` estão obsoletos desde o MySQL 4.0, apenas use `--var=opção`. Veja Seção 5.5.2 [Server parameters], Página 454.

No MySQL 4.0.2 pode-se definir uma variável diretamente com `--variable-name=opção` e `set-variable` não é mais preciso no arquivo de opções.

Se você quiser restringir o valor máximo uma opção de inicialização pode ser definida com `SET`, você pode definí-la usando a opção de linha de comando `--maximum-variable-name`. Veja Seção 5.5.6 [SET OPTION], Página 460.

Note que quando um valor é atribuído a uma variável, o MySQL pode carrigí-lo automaticamente para permanecer dentro de uma faixa dada e também ajusta o valor um pouco para corrigir para o algoritmo usado.

`--safe-mode`
Salta alguns estágios de otimização.

`--safe-show-database`
Com esta opção, o comando `SHOW DATABASES` retorna apenas aqueles bancos de dados para os quais o usuário tem algum tipo de privilégio. Desde a versão 4.0.2 esta opção esta obsoleta e não faz nada (a opção está habilitada por padrão) já que agora temos o privilégio `SHOW DATABASES`. Veja Seção 4.4.1 [GRANT], Página 254.

`--safe-user-create`
Se isto estiver ativo, um usuário não pode criar novos usuários com o comando GRANT, se o usuário não ter o privilégio de `INSERT` na tabela `mysql.user` ou em alguma coluna desta tabela.

`--skip-bdb`
Disabilita o uso de tabelas BDB. Isto economizará memória e pode aumentar a velocidade de algumas operações.

`--skip-concurrent-insert`
Desliga a habilidade de selecionar e inserir ao mesmo tempo em tabelas `MyISAM`. (Isto só é usado se você achar que encontrou um erro neste recurso).

`--skip-delay-key-write;`
No MySQL 4.0.3 você deve usar –delay-key-write=OFF. Ignore a opção `DELAY_KEY_WRITE` para todas as tabelas. Veja Seção 5.5.2 [Parâmetros do servidor], Página 454.

`--skip-grant-tables`
Esta opção faz com que o servidor não use o sistema de privilégio. Isto dá a todos *acesso pleno* a todos os bancos de dados! (Você pode dizer a um servidor em execução para iniciar a usar as tabelas de permissão novamente executando `mysqladmin flush-privileges` ou `mysqladmin reload`.)

`--skip-host-cache`
Nunca utiliza cache para nomes de máquina para resoluções de nomes mais rápidos, mas pesquisa o servidor DNS em todas conexões. Veja Seção 5.5.5 [DNS], Página 459.

`--skip-innodb`
Disabilita o uso de tabelas Innodb. Isto irá economizar memória, espaço em disco e aumentar a velocidade de algumas operações.

`--skip-external-locking (era --skip-locking)`
Não utilizar bloqueio de sistema. Para usar `isamchk` ou `myisamchk` você deve desligar o servidor. Veja Seção 1.2.3 [Stability], Página 8. Perceba que na Versão 3.23 do MySQL pode ser usado `REPAIR` e `CHECK` para reparar/conferir tabelas `MyISAM`.

`--skip-name-resolve`
: Nomes de máquinas não são resolvidos. Todos os valores da coluna `Host` nas tabelas de permissões devem conter números IP ou `localhost`. Veja Seção 5.5.5 [DNS], Página 459.

`--skip-networking`
: Não escutair conexões TCP/IP. Toda interação com `mysqld` deve ser feito através de named pipes ou sockets Unix. Esta opção é altamente recomendada para sistemas onde requisições locais são permitidas. Veja Seção 5.5.5 [DNS], Página 459.

`--skip-new`
: Não utilizar rotinas novas, possívelmente erradas.

`--skip-symlink`
: Opção obsoleta a partir da 4.0.13; use `--skip-symbolic-links` em seu lugar.

`--symbolic-links, --skip-symbolic-links`
: Habilita ou desabilita suporte a link simbólico. Esta opção tem efeitos diferentes no Windows e Unix.

  No Windows, habilitar links simbílicos lhe permite estabelecer um link simbólico a um diretório de banco de dadosi criando um arquivo `directory.sym` que contém o caminho para o diretório real. Veja Seção 5.6.1.3 [Links simbólicos no Windows], Página 468.

  No Unix, habilitar links simbólicos, significa que você pode ligar uma tabela MyISAM ou um arquivo de dados em outro dirtório com as opções `INDEX DIRECTORY` ou `DATA DIRECTORY` da instrução `CREATE TABLE`. Se você deletar ou renomear a tabela, os arquivos para o qual o link simbólico aponta também será deletado/renomeado.

`--skip-safemalloc`
: Se o MySQL é configurado com `--with-debug=full`, todos os programas verificam a memória por erros para cada operação de alocação e liberação de memória. Esta consistência é muito lenta, assim para o servidor você pode evitá-la, quando você não precisar dela usando a opção `--skip-safemalloc`.

`--skip-show-database`
: Não permite o comando 'SHOW DATABASE', a menos que o usuário tenha privilégio **SHOW DATABASES**.

`--skip-stack-trace`
: Não gravar os rastreamentos de pilha. Esta opção é útil quando você estiver executando o `mysqld` sob um depurador. El alguns sistemas você também deve usar esta opção para conseguir um arquivo core. Veja Seção E.1 [Depurando o servidor], Página 1079.

`--skip-thread-priority`
: Desabilita o uso de prioridade das threads para um tempo de resposta mais rápido.

`--socket=path`

No Unix, o arquivo socket para usar em conexões locais no lugar do padrão `/tmp/mysql.sock`. No Windows, o nome do pipe para usar em conexões locais que usam named pipe (padrão `MySQL`).

`--sql-mode=value[,value[,value...]]`

Os valores de opção pode ser qualquer combinação de: `REAL_AS_FLOAT`, `PIPES_AS_CONCAT`, `ANSI_QUOTES`, `IGNORE_SPACE`, `ONLY_FULL_GROUP_BY`, `NO_UNSIGNED_SUBTRACTION`, `NO_AUTO_VALUE_ON_ZERO`, `NO_TABLE_OPTIONS`, `NO_FIELD_OPTIONS`, `NO_KEY_OPTIONS`, `NO_DIR_IN_CREATE`, `MYSQL323`, `MYSQL40`, `DB2`, `MAXDB`, `MSSQL`, `ORACLE`, `POSTGRESQL`, ou `ANSI`. O valor também pode ficar vazio (`--sql-mode=""`) se você desejar limpá-la.

`NO_AUTO_VALUE_ON_ZERO` afeta o tratamento de colunas `AUTO_INCREMENT`. Normalmente, você gera a próxima sequência de números da coluna inserindo `NULL` ou `0` nela. `NO_AUTO_VALUE_ON_ZERO` omite este comportamento para `0`, assim apenas `NULL` gera a próxima sequência de números. Este modo pode ser útil se `0` foi armazenado em uma coluna `AUTO_INCREMENT` da tabela (isto não é recomendado). Por exemplo, se você fizer um dumpo de uma tabela com `mysqldump` e então recarregá-la, normalmente o MySQL ira gerar uma nova sequência de números quando encontrar valores `0`, resultando em uma tabela com conteúdo diferente daquele do qual foi feito o dump. Habilitando `NO_AUTO_VALUE_ON_ZERO` antes de recarregar o arquivo de dump soluciona este problema. (A partir do MySQL 4.1.1, quando este valor se tornar disponível, o `mysqldump` inclui automaticamente a saída do dump para habilitar `NO_AUTO_VALUE_ON_ZERO`.)

Diversos dos valores de opção são usados para compatibilidade com outros servidores. Se especificado, eles fazer o servidor omitir da saída de `SHOW CREATE TABLE` aquelas partes da instrução que não são entendidas pelas versões anteriores do MySQL ou outros servidores de banco de dados. Usar estes valores de opções resulta em instruções `CREATE TABLE` que são mais portáveis para usar com outros servidores:

- Os valores `NO_TABLE_OPTIONS`, `NO_FIELD_OPTIONS`, `NO_DIR_IN_CREATE`, e `NO_KEY_OPTIONS` causam a omissão da tabela de opções, ou opções pertencentes a definição de colunas ou índices.
- Os valroes `MYSQL323` e `MYSQL40` são para compatibilidade com o MySQL 3.23 e MySQL 4.0.
- O valor usado para compatibilidade com outros servidores são `DB2`, `MAXDB`, `MSSQL`, `ORACLE`, e `POSTGRESQL`.

Estas opções também afetam a saída do `mysqldump`, porque este programa usa `SHOW CREATE TABLE` para obter a instrução de criação da tabela a qual ele inclue em sua própria saída.

Diversos valores de opções podem ter um efeito complexo porque eles são atalhos para um grupo ou conjunto de valores. Por exemplo, você pode dizer ao servidor para executar em modo ANSI usando a opção `--sql-mode=ansi` (ou `--ansi`), que é equivalente a especificar ambas das seguintes opções de linhas de comando:

```
--sql-mode=REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ONLY_FU
--transaction-isolation=SERIALIZABLE
```

Note que especificar o modo ANSI desta forma também tem o efeito de configurar o nível de isolação da transação.

Para mais informações sobre executar o servidor em modo ANSI, veja Seção 1.8.2 [ANSI mode], Página 42.

Outros valores de "grupos" são `DB2`, `MAXDB`, `MSSQL`, `ORACLE`, e `POSTGRESQL`. Esepcificar qualquer um dele ativa os valores `PIPES_AS_CONCAT`, `ANSI_QUOTES`, `IGNORE_SPACE`, `NO_TABLE_OPTIONS`, `NO_FIELD_OPTIONS`, e `NO_KEY_OPTIONS`.

A opção `--sql-mode` foi adicionada no MySQL 3.23.41. O valor `NO_UNSIGNED_SUBTRACTION` foi adicionado na versão 4.0.0. `NO_DIR_IN_CREATE` foi adicionado na versão 4.0.15. `NO_AUTO_VALUE_ON_ZERO`, `NO_TABLE_OPTIONS`, `NO_FIELD_OPTIONS`, `NO_KEY_OPTIONS`, `MYSQL323`, `MYSQL40`, `DB2`, `MAXDB`, `MSSQL`, `ORACLE`, `POSTGRESQL`, e `ANSI` foram adicionados na versão 4.1.1.

`--temp-pool`

Usar esta opção fará com que a maioria dos arquivos temporários criados pelo servidor para usarem um pequeno conjunto de nomes, em vez de um único nome para cada novo arquivo. Isto é para contornar um problema no kernel do Linux ao tratar com a criação de muitos arquivos novos com nomes diferentes. Com o comportamento antigo, o Linux parece ter "perda" de memória, já que ela é alocada na cache de entrada do diretório em vez da cache de disco.

`--transaction-isolation={ READ-UNCOMMITTED | READ-COMMITTED | REPEATABLE-READ | SERIALIZABLE }`

Configura o nível de isolação da transação padrão. Veja Seção 6.7.6 [`SET TRANSACTION`], Página 619.

`-t, --tmpdir=path`

Caminho do diretório usado para criar os arquivos temporários. Ele pode ser útil se o seu diretório padrão `/tmp` está em uma partição muito pequena para armazenar tabelas temporárias. A partir do MySQL 4.1, esta opção aceita diversos caminhos usados do modo round-robin. Os caminhos devem ser separados por dois pontos (`:`) (ponto e vírgula (`;`) no Windows). Eles serão usados de acordo com o método round-robin.

`-u, --user=[nome_usuário | id_usuário]`

Executar o servidor `mysqld` como o usuário `nome_usuário` ou `id_usário` (numérica). ("User" neste contexto se refere a conta de login do sistema, não um usuário MySQL listado na tabela de permissões.)

Esta opção é *obrigatória* quando o `mysqld` é iniciado como usuário `root`. O servidor irá alterar o ID do usuário durante sua inicialização, fazendo com que ele seja executado como este usuário particular em vez de `root`. Veja Seção 4.3.2 [Security], Página 229.

A partir do MySQL 3.23.56 e 4.0.12: Para evitar um possível furo na segurança onde um usuário adiciona uma opção `--user=root` a algum arquivo '`my.cnf`' (fazendo o servidor executar como `root`, o `mysqld` usa apenas a primeira opção `--user` especificada e produz um aviso se houver múltiplas opções `--user`.

As opções em '`/etc/my.cnf`' e '`datadir/my.cnf`' são processadas antes de uma opção de linha de comando, assim é recomendado que você coloque uma opção

`--user` em '`/etc/my.cnf`' e especifique um outro valor diferente de `root`. A opção em '`/etc/my.cnf`' será encontrada antes de qualques outra opção `--user`, o que assegura que o servidor não execute como `root`, e que um aviso seja exibido se qualquer outra opção `--user` for encontrada.

`-V, --version`
Mostra a informação da versão e sai.

`-W, --log-warnings`
Imprime avisos como `Aborted connection...` no arquivo `.err`. É recomendável habilitar esta opção, por exemplo, se você estiver usando replicação (você obterá a mensagem sobre o que está acontecendo como falhas de rede e reconexões). Veja Seção A.2.10 [Erros de comunicação], Página 918.

Esta opção se chamava `--warnings`.

Pode se alterar a maioria dos valores de um servidor em execução com o comnado `SET`. Veja Seção 5.5.6 [`SET OPTION`], Página 460.

4.1.2 Arquivo de Opções '`my.cnf`'

O MySQL pode, desde a versão 3.22, ler as opções padrões de inicialização para o servidor e para clientes dos arquivos de opções.

No Windows, o MySQL lê opções padrões dos seguintes arquivos:

Nome do Arquivo	**Propósito**
`Windows-directory\my.ini`	Opções globais
`C:\my.cnf`	Opções globais

`Windows-directory` é a localização do seu diretório Windows.

No Unix, o MySQL lê opções padrões dos seguintes arquivos:

Nome do arquivo	**Propósito**
`/etc/my.cnf`	Opções globais
`DATADIR/my.cnf`	Opções específicas do servidor
`defaults-extra-file`	O arquivo especificado com `--defaults-extra-file=#`
`~/.my.cnf`	Opções específicas do usuário

`DATADIR` é o diretório de dados do MySQL (normalmente '`/usr/local/mysql/data`' para instalações binárias ou '`/usr/local/var`' para instalações de código fonte). Perceba que este é o diretório que foi especificado na hora da configuração, não o especificado com `--datadir` quando o `mysqld` inicia! (`--datadir` não tem efeito sobre o local onde o servidor procura por arquivos de opções, porque ele procura pelos arquivos antes de processar qualquer argumento da linha de comando.)

Note que no Windows, você deve especificar todos os caminhos no arquivo de opção com `/` no lugar de `\`. Se for utilizado o `\`, será necessário digitá-lo duas vezes, pois o `\` é o caractere de escape no MySQL.

O MySQL tenta ler os arquivos de opções na ordem listada acima. Se múltiplos arquivos de opções existirem, uma opção especificada em um arquivo lido depois recebe a precedência sobre a mesma opção especificada em um arquivo lido anteriormente. Opções especificadas

na linha de comando recebem a precedência sobre opções especificadas em qualquer arquivo de opções. Algumas opções podem ser especificadas usando variáveis de ambiente. Opções especificadas na linha de comando ou nos arquivos de opção tem precendencia sobre valores nas variáveis de ambiente. Veja Apêndice F [Variáveis de ambiente], Página 1092.

Os seguintes programas suportam arquivos de opções: `mysql`, `mysqladmin`, `mysqld`, `mysqld_safe`, `mysql.server`, `mysqldump`, `mysqlimport`, `mysqlshow`, `mysqlcheck`, `myisamchk`, e `myisampack`.

Desde a versão 4.0.2, você pode usar o prefixo `loose` para opções de linha de comando (ou opções no my.cnf). Se uma opção possui o prefixo `loose`, o programa que a ler não finalizará com um erro se uma opção for desconhecida, mas apenas enviará um aviso:

```
shell> mysql --loose-no-such-option
```

Você pode usar arquivos de opções para especificar qualquer opção extendida que o programa suporte! Execute o programa com `--help` para obter uma lista das opções disponíveis.

Um arquivo de opções pode conter linhas na seguinte forma:

`#comentario`
: Linhas de comentário iniciam com o caractere '`#`' ou '`;`'. Comentários podem iniciar no meio de uma linha também. Linhas vazias são ignoradas.

`[grupo]`
: `grupo` é o nome do programa ou grupo para o qual você irá configurar as opções. Depois de uma linha de grupo, qualquer linha de `opção` ou `set-variable` são referentes ao grupo até o final do arquivo de opções ou outra linha de início de grupo.

`opção`
: Isto é equivalente à `--opção` na linha de comando.

`opção=valor`
: Isto é equivalente à `--opção=valor` na linha de comando. Por favor, note que você deve colocar um argumento entre aspas duplas, se o argumento de uma opção conter um caracter de comentário.

`set-variable = nome=valor`
: Isto é equivalente à `--set-variable nome=valor` na linha de comando.

  Por favor, notem que `--set-variable` está obsoleto desde o MySQL 4.0; a partir desta versão os nomes das variáveis de programa podem ser usados como nome de opções. Na linha de comando, use apenas `--nome=valor`. Em um arquivo de opção, use `nome=valor`.

O grupo `[client]` permite especificar opções para todos clientes MySQL (não o `mysqld`). Este é o grupo perfeito de se usar para espeficar a senha que você usa para conectar ao servidor. (Mas tenha certeza que o arquivo de opções só pode ser lido e gravado por você)

Se você quiser criar opções que devem ser lidas por uma versão específica do servidor `mysqld` você pode fazer isto com `[mysqld-4.0]`, `[mysqld-4.1]` etc:

```
[mysqld-4.0]
new
```

A nova opção acima só será usada com o versões 4.0.x do servidor MySQL.

Perceba que para opções e valores, todos espaços em branco são automaticamente apagados. Você pode usar a sequencia de escape '`\b`', '`\t`', '`\n`', '`\r`', '`\\`' e '`\s`' no valor da string ('`\s`' == espaço).

Aqui está um típico arquivo de opções globais.

```
[client]
port=3306
socket=/tmp/mysql.sock

[mysqld]
port=3306
socket=/tmp/mysql.sock
set-variable = key_buffer_size=16M
set-variable = max_allowed_packet=1M

[mysqldump]
quick
```

Aqui está um típico arquivo de opções do usuário

```
[client]
# A senha seguinte será enviada para todos clientes MySQL
password="minha_senha"

[mysql]
no-auto-rehash
set-variable = connect_timeout=2

[mysqlhotcopy]
interactive-timeout
```

Se você tem uma distribuição fonte, você encontrará arquivos de exemplo de configuração chamados '`my-xxxx.cnf`' no diretório '`support-files`'. Se você tem uma distribuição binária olhe no diretório de instalação '`DIR/support-file`', onde `DIR` é o caminho para o diretório de instalação (normalmente '`C:\mysql`' ou '`/usr/local/mysql`'). Atualmente existem arquivos de configuração para sistemas pequenos, médios, grandes e enormes. Você pode copiar '`my-xxxx.cnf`' para seu diretório home (renomeie a cópia para '`.my.cnf`' para experimentar.

Todos os programas MySQL que suportam arquivos de opções aceitam opções:

Opção	**Descrição**
`--no-defaults`	Não lê nenhum arquivo de opções.
`--print-defaults`	Imprima o nome do programa e todas opções.
`--defaults-file=caminho-para-arquivo-padrão`	Utilize somente o arquivo de configuração específicado.
`--defaults-extra-file=caminho-para-arquivo-padrão`	Leia este arquivo de configuração depois do arquivo de configuração global mas antes do arquivo de configuração do usuário.

Perceba que as opções acima devem vir primeiro na linha de comando para funcionar, com exceção que `--print-defaults` deve ser usado logo depois dos comandos `--defaults-file` ou `--defaults-extra-file`.

Notas para desenvolvedores: O tratamento de arquivos de opções é implementado simplesmente processando todos as opções coincidentes (isto é, opções no grupo apropriado) antes

de qualquer argumento da linha de comando. Isto funciona bem para programas que usam a última instância de uma opção que é especificada diversas vezes. Se você tem um programa antigo que trata opções especificadas várias vezes desta forma mas não lê arquivos de opções, você só precisa adicionar duas linhas para lhe dar esta capacidade. Verifique o código fonte de qualquer um dos clientes MySQL padrão para ver como fazer isto.

Nos scripts shell você pode usar o comando '`my_print_defaults`' para analisar os arquivos de opção. O seguinte exemplo mostar a saída que `my_print_defaults` pode produzir quando quando pedido para mostrar as opções encontradas nos grupos `[client]` e `[mysql]`:

```
shell> my_print_defaults client mysql
--port=3306
--socket=/tmp/mysql.sock
--no-auto-rehash
```

4.2 Executando Múltiplos MySQL Servers na Mesma Máquina

Em alguns casos você pode precisar de executar múltiplos servidores `mysqld` executando na mesma máquina. Você pode desejar testar uma nova versão do MySQL enquanto a deixa a sua instalação da versão de produção existente sem perturbação. Ou você pode desejar dar acesso a diferentes usuários em diferentes servidores `mysqld` gerenciados por eles mesmos. (Por exemplo, você pode seu um provedor de serviços de internet que quer fornecer instalações independentes do MySQL para clientes diferentes).

Para executar múltiplos servidores em uma única máquina, cada servidor deve ter valores únicos para diversos parâmetros operacionais. Isto deve ser configurado na linha de comando ou em arquivos de opções. Veja Seção 4.1.1 [Command-line options], Página 207 e Seção 4.1.2 [Option files], Página 216.

Pelo menos as seguintes opções devem ser diferente para cada servidor:

- `--port=port_num`
- `--socket=path`
- `--shared-memory-base-name` (apenas Windows; novo no MySQL 4.1)
- `--pid-file=path` (apenas Unix)

`--port` controla o número da porta para conexões TCP/IP. `--socket` controla o caminho do arquivo de socket no Unix e o nome do named pipe no Windows. (É necessário nomes de pipes distintos no Windows apenas para aqueles servidores que suportam conexão named pipes.)

`--shared-memory-base-name` designa o nome da memória compartilhada por um servidor Windows para permitir que o cliente se conecte via memória compartilhada. `--pid-file` indice o nome do arquivo no qual o Unix gravar a ID do seu processo.

Se você usar as seguintes opções, elas deve ser diferentes para cada servidor:

- `--log=path`
- `--log-bin=path`
- `--log-update=path`

- `--log-error=path`
- `--log-isam=path`
- `--bdb-logdir=path`

Se você quiser mais desempenho, você também pode especificar as seguinte opções diferentemente para cada servidor para distribuir a carga entre vários discos físicos:

- `--tmpdir=path`
- `--bdb-tmpdir=path`

Normalmente, cada servidor também deve usar um diretório de dados diferentes, que é especificado usando a opção `--datadir=path`.

AVISO: Normalmente você nunca deve ter dois servidores que atualizam dados no mesmo banco de dados! Isto pode levar a supresas inesperadas se seu o seu sistema operacionalnão suporta lock de sistema a prova de falhas, isto pode provocar surpresas indesejáveis! Se (apesar deste aviso) você executar diversos servidores usando o mesmo diretório de dados e eles tiverem com o log habilitado, você usar as opções apropriadas para especificar os nomes dos arquivos de log que são únicos em cada servidor. Senão, o servidores podem tentar gravar no mesmo arquivo de log.

Este aviso contra o compartilhamento de arquivos de dados entre servidores também se aplica em um ambeinte NFS. Permitir vários servidores MySQL acessarem um diretório de dados comum sobre NFS, é normalmente uma **MÁ IDÉIA**!

- O primeiro problema é que o `NFS` se tornará um gargalo, tornando o sistema lento. Ele não se destina para este tipo de uso.
- Outro risco com NFS é que você terá que conseguir um modo de se certificar que dois ou mais servidores não estão interferindo uns com os outros. Normalmente o lock de arquivo é tratado pelo daemon `lockd`, mas no momento não existe nenhuma plataforma que fara o locck 100% de segurança, em todas as situações.

Facilite a sua vida: esqueça sobre compartilhar um diretório de dados entre servidores sobre NFS. A solução melhor é ter um computador com um sistema operacional que manipule threads de forma eficiente threads e tenha diversas CPUs nele.

Se você tiver múltiplas instalações do MySQL em diferentes locais, normalemente você pode especificar o diretório de instalação base de cada servidor com a opção `--basedir=caminho` para fazer que cada servidor use diferentes diretórios de dados, arquivos de log e arquivos PID. (O padrão para todos estes valores são determinados em relação ao diretório base.) Neste caso, as únicas outras opções que você precisa especificar são as opções `--socket` e `--port`. Por exempo, suponha que você instalou a versão binária do MySQL (arquivos '`.tar`') em diferentes locais, assim você pode iniciar o servidor usando o comando `./bin/mysqld_safe` sob o diretório base correspondente de cada instalação. `mysqld_safe` determinará a opção `--basedir` apropriada para passar para `mysqld`, e você precisa especificar apenas as opções `--socket` e `--port` para o `mysqld_safe`.

Como discutido nas seções a seguir, é possível iniciar servidores adicionais configurando variáveis de ambiente ou especificando as opções de linha de comando apropriada. No entanto, se você precisa executar múltiplos servidores em uma base mais permanente, será mais coonveniente usar os arquivos de opções para especificar, para cada servidor, aquelas opções que devem ser únicas para ele. Veja Seção 4.1.2 [Option files], Página 216.

4.2.1 Executando Múltiplos Servidores no Windows

Você pode executar múltiplos servidor no Windows iniciando-os manualmente a partir da linha de comando, cada um com o parâmetro operacional apropriado. Em sistemas baseados no Windows NT, você também tem a opção de instalar vários servidores como serviços Windows e executá-los deste modo. Instruções gerais sobre a execucão de servidores MySQL a partir da linha de comando ou como serviços são dados em Seção 2.6.1 [Windows], Página 133. Esta seção descreve como se certificar de que você inicioou cada servidor com valores diferentes para aquelas opções de inicialização que devem ser unicas por servidor, como o diretório de dados. (Estas opções são descritas em Seção 4.2 [Multiple servers], Página 219.)

4.2.1.1 Iniciando Múltiplos Servidores na Linha de Comando

Para iniciar vários servidores manualmente na linha de comando, você pode especificar a opção apropriada na linha de comando ou no arquivo de opções. É mais conveniente colocar as opções em um arquivo de opção. Para fazer isto, crie uma arquivo de opção para cada servidor e mostre ao servidor o nome do arquivo com a opção `--defaults-file` quando você executá-lo.

Suponha que você queira executar o `mysqld` na porta 3307 com um diretório de dados de '`C:\mydata1`', e `mysqld-max` na porta 3308 com um diretório de dados de '`C:\mydata2`'. Para conseguir isto, crie dois arquivos de opções. Por exemplo, crie um arquivo chamado '`C:\my-opts1.cnf`' que se pareça com isto:

```
[mysqld]
datadir = C:/mydata1
port = 3307
```

Crie um segundo arquivo chamado '`C:\my-opts2.cnf`' que se pareça com isto:

```
[mysqld]
datadir = C:/mydata2
port = 3308
```

Então inicie cada servidor com seus próprios arquivos de opção:

```
shell> mysqld --defaults-file=C:\my-opts1.cnf
shell> mysqld-max --defaults-file=C:\my-opts2.cnf
```

(No NT, o servidor iniciará em segundo plano, assim você precisará enviar estes dois comandos em janelas de console separadas.)

Para desligar o servidor, você deve conectar a porta apropriada:

```
shell> mysqladmin --port=3307 shutdown
shell> mysqladmin --port=3308 shutdown
```

Servidores configurados como descrito permitirá que clientes se conectem por TCP/IP. Se você também quiser permitir conexões named pipe, use os servidores `mysqld-nt` ou `mysqld-max-nt` e especifique as opção que habilitem o named pipe e especifique os seus nomes. (Cada servidor que suporta conexões named pipes deve ter um nome único). Por exemplo, o arquivo '`C:\my-opts1.cnf`' pode ser escrito da seguinte maneira:

```
[mysqld]
datadir = C:/mydata1
port = 3307
enable-named-pipe
socket = mypipe1
```

Estão inicie o servidor desta forma:

```
shell> mysqld-nt --defaults-file=C:\my-opts1.cnf
```

'`C:\my-opts2.cnf`' seria modificado de forma parecida para uso com o segundo servidor.

4.2.1.2 Iniciando Múltiplos Servidores Como Serviços

Em sistemas baseados no NT, um servidor MySQL pode ser executado como um serviço Windows. O procedimento para instalação, controle e remoção de um único serviço MySQL está descrito em Seção 2.1.1.7 [NT start], Página 66.

A partir do MySQL 4.0.2, você pode instalar vários servidores como serviços. Neste caso, você deve ter certeza de que cada servidor usa um nome de serviço diferente junto com todos os outros parâmetros que devem ser único por servidor.

Para as seguintes instruções, assuma que você queira executar o servidor `mysqld-nt` a partir de duas versões diferentes do MySQL que está instalado em '`C:\mysql-4.0.8`' e '`C:\mysql-4.0.17`', respectivamente. (Este pode ser o caso se você estiver executando a versão 4.0.8 como seu servidor de produção, mas queira testar o 4.0.17 antes de atualizá-lo.)

Os seguintes princípios são relevantes ao instalr um serviço MySQL com a opção `--install`:

- Se você não especificar o nome do serviço, o servidor usa o nome padrão do serviço (`MySQL`) e o servidor lê as opções do grupo `[mysqld]` no arquivo de opções padrão.
- Se você especificar um nome de serviço depois da opção `--install`, o servidor ignora o grupo de opção `[mysqld]` e lê as opções do grupo que tem o mesmo nome que o serviço. O servidor lê as opções do arquivo de opção padrão.
- Se você especificar uma opção `--defaults-file` depois do nome do serviço, o servidor ignora o arquivo de opções padrão e lê as opções apenas do grupo `[mysqld]` do arquivo chamado.

Este princípios também se aplicam se você intalar um servidor usando a opção `--install-manual`.

Baseado na informação anterior, você tem diversos de configurar vários serviços. As seguintes instruções descrevem alguns exemplos. Antes de tentar qualquer uma delas esteja certo de que você desligou e removeu qualquer serviço MySQL existente primeiro.

- Especifique as opções para todos os serviços em um dos arquivos de opções padrão. Para fazer isto, use um nome de serviço diferente para cada servidor. Suponha que você queira executar o `mysqld-nt` 4.0.8 usando o nome de serviço `[mysqld1]` e o `mysqld-nt` 4.0.17 usando o nome de serviço `mysqld2`. Neste caso você pode usar o grupo `[mysqld1]` para o 4.0.8 e o grupo `[mysqld2]` para o MySQL 4.0.14. Por exemplo, você pode configurar o '`C:\my.cnf`' desta forma:

  ```
  # opções para o serviço mysqld1
  [mysqld1]
  ```

```
basedir = C:/mysql-4.0.8
port = 3307
enable-named-pipe
socket = mypipe1

# opções para o serviço mysql2
[mysqld2]
basedir = C:/mysql-4.0.17
port = 3308
enable-named-pipe
socket = mypipe2
```

Instale os serviços como a seguir, usando o caminho completo para o servidor para assegurar que o Windows registra o programa executável correto para cada serviço:

```
shell> C:\mysql-4.0.8\bin\mysqld-nt --install mysqld1
shell> C:\mysql-4.0.17\bin\mysqld-nt --install mysqld2
```

Para iniciar os serviços, use o gerenciador de serviços, ou use `NET START` com o nome de serviço apropriado:

```
shell> NET START mysqld1
shell> NET START mysqld2
```

Para parar os serviços, use o gerenciador de serviços, ou use `NET STOP` com o mesmo nome de serviço.

```
shell> NET STOP mysqld1
shell> NET STOP mysqld2
```

Nota: Antes do MySQL 4.0.17, apenas um servidor instalado usando o nome de serviço padrão (`MySQL`) ou instalado com um nome de serviço de `mysqld` irá ler o grupo `[mysqld]` no arquivo de opções padrão. A partir da versão 4.0.17, todos os servidores lêem o grupo `[mysqld]` se eles lêem o arquivo de opções padrão, mesmo de esles estão instalados usando outro nome de serviço. Isto permite que você use o grupo `[mysqld]` para opções que devam ser usadas por todos os serviços MySQL, e um grupo de opção com o nome de cada serviço para o uso do servidor com aquele nome de serviço.

- Especifique as opções para cada servidor em arquivos separados e use `--defaults-file` quando instalar os serviços para dizer para cada servidor que arquivo usar. Neste caso, cada arquivo deve listar as opções usando um grupo `[mysqld]`.

  Com esta abordagem, para especificar as opções para o `mysqld-nt` 4.0.8, crie um arquivo '`C:\my-opts1.cnf`' que se pareça com:

  ```
  [mysqld]
  basedir = C:/mysql-4.0.8
  port = 3307
  enable-named-pipe
  socket = mypipe1
  ```

  Para o `mysqld-nt` 4.0.17, crie um arquivo '`C:\my-opts2.cnf`' que se pareça com:

  ```
  [mysqld]
  basedir = C:/mysql-4.0.17
  port = 3308
  ```

```
enable-named-pipe
socket = mypipe2
```

Instale o serviço como indicado a seguir (digite cada comando em uma única linha):

```
shell> C:\mysql-4.0.8\bin\mysqld-nt --install mysqld1
           --defaults-file=C:\my-opts1.cnf
shell> C:\mysql-4.0.17\bin\mysqld-nt --install mysqld2
           --defaults-file=C:\my-opts2.cnf
```

Para usar uma opção `--defaults-file` quando instalar um servidor MySQL como um serviço, você deve anteceder a opção com o nome do serviço.

Depois de instalarm, inicie e para os serviços do mesmo modo que no exemplo anterior.

Para remover vários serviços, use `mysqld --remove` para cada um, especificando um nome de serviço depois da opção `--remove` se o serviço a ser removido tiver um nome difertente do padrão.

4.2.2 Executando Múltiplos Servidores no Unix

O modo mais fácil de executar diversos servidores no Unix é compilá-los com diferentes portas TCP/IP e arquivos socket, assim cada um está escutando em diferentes interfaces de rede. Também, compilando em diferentes diretórios bases para instalação, que automaticamente resulta em diferentes localizações de diretórios de dados, arquivos log e arquivos PID para cada um dos seus servidores.

Considere que um servidor existente está configurado para a porta e arquivo socket padrões. Para configurar um novo servidor para ter parâmetros operacionais diferentes, use um comando `configure` assim:

```
shell> ./configure --with-tcp-port=port_number \
            --with-unix-socket-path=nome_arquivo \
            --prefix=/usr/local/mysql-4.0.17
```

Aqui `número_porta` e `nome_arquivo` deve ser diferente que o número da porta e o caminho do arquivo socket padrões e o valor `--prefix` deve especificar um diretório de instalação diferente daquele usado pelo servidor existente.

Você pode conferir o socket usado por qualquer servidor MySQL em execução com este comando:

Se você tem um servidor MySQL escutando em uma porta dada, você pode usar o seguinte comando para descobrir quaie parâmetros operacionais ele está usando para diversas variáveis importantes configuráveis, incluíndo o diretório base e o nome do socket:

```
shell> mysqladmin --host=host_name --port=port_number variables
```

Com a informação exibida por aquele comando, você pode dizer quais valores de opção **não** usar ao configurar um servidor adicional.

Note que se você especificar "`localhost`" como o nome da máquina, `mysqladmin` irá por padrão usar uma conexão sockets Unix em vez de TCP/IP. No MySQL 4.1 você também pode especificar o protocolo a ser usado com a opção `--protocol={TCP | SOCKET | PIPE | MEMORY}`.

Não é necessário compilar um novo servidor MySQL apenas para iniciar com uma arquivo socket ou número de porta TCP/IP diferentes. Também é possível especificar estes valores em tempo de execução. Um modo de fazê-lo é usando as opções de linha de comando:

```
shell> /path/to/mysqld_safe --socket=file_name --port=port_number
```

Para usar outro diretório de banco de dados para o segundo servidor, passe uma opção `--datadir=caminho` para o `mysqld_safe`.

Um outro modo de conseguir este efeito é usar as variáveis de ambiente para configurar o nome do socket e o número da porta:

```
shell> MYSQL_UNIX_PORT=/tmp/mysqld-new.sock
shell> MYSQL_TCP_PORT=3307
shell> export MYSQL_UNIX_PORT MYSQL_TCP_PORT
shell> scripts/mysql_install_db
shell> bin/mysqld_safe &
```

Este é um modo rápido para iniciar um segundo servidor para teste. O bom deste método é que a configuração das variáveis de ambiente se aplicarão a qualquer programa cliente que você chame da shell acima. Assim, as conexões para estes clientes serão automaticamente direcionadas para o segundo servidor!

Apêndice F [Variáveis de ambiente], Página 1092 inclue uma lista de outras variáveis de ambiente que você pode usar e que afetam o `mysqld`.

Para a execução automatica do servidor, seu script de inicialização que é executado no tempo de boot deve executar o seguinte comando uma vez para cada servidor com um caminmho apropriado do arquivo de opção para cada comando:

```
mysqld_safe --defaults-file=path-to-option-file
```

Cada arquivo de opção deve conter valores específicos para um dados servidor.

No Unix, o script `mysqld_multi` é outro modo de de iniciar vários servidores. Veja Seção 4.8.3 [`mysqld_multi`], Página 333.

4.2.3 Usando Programas Clientes em um Ambiente Multi-Servidor

Quando você quiser conectar com um programa cliente a um servidor MySQL que está escutando diferentes interfaces de rede em vez daquelas compiladas em seu programa cliente, você pode conectar usando um dos seguintes métodos:

- Inicie o cliente com `--host=nome_máquina --port=número_porta` para conectar com TCP/IP a uma máquina remota, ou com `--host=localhost --socket=nome_arquivo` para conectar a máquina local via um socket Unix ou um named pipe do Windowes.
- No MySQL 4.1, inicie o cliente com `--protocol=tcp` para conectar via TCP/IP, `--protocol=socket` para conectar via socket Unix ou `--protocol=pipe` para conectar via named pipe, ou `--protocol=memory` para conectar via memória compartilhada. Para conexões TCP/IP, você também pode precisar especificar as opções `--host` e `--port`. Para outros tipos de conexões, você pode precisar especificar uma opção `--socket` para definir um nome de socket ou named pipe name, ou uma opção `--shared-memory-base-name` para especificar o nome da memória compartilhada.

No Unix, configure as variáveis de ambiente `MYSQL_UNIX_PORT` e `MYSQL_TCP_PORT` para apontar para o socket Unix e porta TCP/IP antes de iniciar seus clientes. Se você normalmente utiliza uma porta ou socket específico, você pode colocar os comandos para configurar as variáveis de ambiente no arquivo '`.login`', assim eles serão aplicados sempre quer você logar no sistema. Veja Apêndice F [Environment variables], Página 1092.

- Especifique o socket e porta TCP/IP padrões no grupo `[clients]` de um arquivo de opções. Por exemplo, você pode usar '`C:\my.cnf`' no WIndows ou o arquivo '`.my.cnf`' em seu diretório home no Unix. Veja Seção 4.1.2 [Option files], Página 216.
- Em um programa C, você pode especificar os argumentos de porta ou socket na chamada de `mysql_real_connect()`. Você também pode ter o programa lendo de um arquivo de opções chamando `mysql_options()`. Veja Seção 12.1.3 [C API functions], Página 783.
- Se você estiver usando o módulo Perl `DBD::mysql` você pode ler as opções dos arquivos de opções do MySQL. Por exemplo:

```
$dsn = "DBI:mysql:test;mysql_read_default_group=client;"
        . "mysql_read_default_file=/usr/local/mysql/data/my.cnf";
$dbh = DBI->connect($dsn, $user, $password);
```

  Veja Seção 12.5.2 [Perl DBI Class], Página 881.

4.3 Detalhes Gerais de Segurança e o Sistema de Privilégio de Acesso do MySQL

O MySQL tem um sistema de segurança/privilégios avançado mas não padrão. A próxima seção descreve como ele funciona.

4.3.1 Segurança Geral

Qualquer um usando o MySQL em um computador conectado à internet deve ler esta seção para evitar os erros de segurança mais comuns.

Discutindo segurança, nós enfatizamos a a necessidade de proteger completamente o servidor (não simplesmente o servidor MySQL) contra todos os tipos de ataques aplicáveis: eavesdropping, altering, playback e denial of service. Não cobriremos todos os aspectos de disponibilidade e tolerância a falhas aqui.

O MySQL utiliza a segurança baseado em Listas de Controle de Acesso (ACL) para todas conexões, consultas e outras operações que um usuário pode tentar realizar. Existe também algum suporte para conexões criptografadasSSL entre clientes MySQL e servidores. Vários dos conceitos discutidos aqui não são específicos do MySQL; as mesmas idéias podem ser aplicadas para a maioria das aplicações.

Quando executando o MySQL, siga estes procedimentos sempre que possível:

- **nunca conceda a alguém (exceto ao usuário root do mysql) acesso à tabela `user` no banco de dados `mysql`!**. Isto é perigoso. **A senha criptografada é a senha real no MySQL**. Se você conhece a senha listada na tabela `user` para um determinado usuário, você **pode facilmente logar como este usuário se tiver acesso à máquina relacionada para aquela conta**.

- Aprenda o sistema de controle de acessos do MySQL. Os comandos `GRANT` e `REVOKE` são usados para controlar o acesso ao MySQL. Não conceda mais privilégios do que o necessário. Nunca conceda privilégios para todas as máquinas.

  Checklist:

  – Tente `mysql -u root`. Se você conseguir conectar com sucesso ao servidor sem a solicitação de uma senha, você tem problemas. Qualquer um pode conectar ao seu servidor MySQL como o usuário `root` com privilégios plenos! Revise as instruções de instalação do MySQL, prestando atenção particularmente ao item sobre configuração da senha do usuário `root`.

  – Utilize o comando `SHOW GRANTS` e confira para ver quem tem acesso a o que. Remova aqueles privilégios que não são necessários utilizando o comando `REVOKE`.

- Não mantenha nenhuma senha de texto puro no seu banco de dados. Quando seu computador fica comprometido, o intruso pode obter a lista completa de senhas e utilizá-las. Utilize a função `MD5()`, `SHA1()` ou qualquer função de embaralhamento de via única.

- Não escolha senhas de dicionários. Existem programas especiais para quebrá-las. Mesmo senhas como "xfish98" não sao boas. Muito melhor seria "duag98" que contém a mesma palavra 'fish mas digitada uma letra a esquerda em um teclado QWERTY convencional. Outro método seria usar "Mhall" que é obtido dos primeiros caracteres de cada palavra na frase "Mary has a litle lamb". Isto é fácil de lembrar e digitar, mas dificulta que alguém que não a conheça a advinhe.

- Invista em um firewall. Ele protege você de pelo menos 50% de todos os tipos de exploits em qualquer software. Coloque o MySQL atrás do firewall ou em uma zona desmilitarizada (DMZ).

  Checklist:

  – Tente examinar suas portas da Internet utilizando alguma ferramenta como o `nmap`. O MySQL utiliza a porta 3306 por padrão. Esta porta não deve ser acessível para máquinas não confiáveis. Outra maneira simples para conferir se sua porta do MySQL está aberta ou não é tentar o seguinte comando de alguma máquina remota, onde `nome_máquina` é o nome da máquina ou o endereço IP de seu servidor MySQL:

    ```
    shell> telnet nome_máquina 3306
    ```

    Se você obter uma conexão e alguns caracteres, a porta está aberta e deve ser fechada no seu firewall ou roteador, a menos que você realmente tenha uma boa razão para mantê-la aberta. Se o `telnet` apenas parar ou a conexão for recusada, tudo está bem; a porta está bloqueada.

- Não confie em nenhum dado incluídos pelos seus usuários. Eles podem tentar enganar seu código entrando com caracteres especiais ou sequencias de escape nos formulários Web, URLS ou qualquer aplicação que você construa. Tenha certeza que sua aplicação continua segura se um usuário entrar com algo do tipo "`; DROP DATABASE mysql;`". Este é um exemplo extremo, mas grandes falhas de segurança ou perda de dados podem ocorrer como o resultado de hackers utilizando técnicas similares, se você não estiver preparado para eles.

  Também lembre de conferir dados numéricos. Um erro comum é proteger somente as strings. Em alguns casos as pessoas pensam que se um banco de dados contém somente

dados disponíveis publicamente, ele não precisa ser protegido. Isto não é verdade. No mínimo ataques do tipo denial-of-service podem ser feitos nestes bancos de dados. A maneira mais simples para proteger deste tipo de ataque é usar apóstrofos em torno das contantes numéricas: `SELECT * FROM tabela WHERE ID='234'` em vez de `SELECT * FROM table WHERE ID=234`. O MySQL automaticamente converte esta string para um número e corta todos os símbolos não-numéricos dela.

Checklist:

- Todas aplicações Web:
  - Tente inserir '' e '"' em todos seus formulários Web. Se você obter qualquer tipo de erro do MySQL, investigue o problema imediatamente.
  - Tente modificar qualquer URL dinâmica adicionando `%22` ('"'), `%23` ('#') e `%27` ('') na URL.
  - Tente modificar os tipos de dados nas URLs dinâmicas de numérico para caractere contendo caracteres dos exemplos anteriores. Sua aplicação deve ser segura contra estes ataques e similares.
  - Tente inserir caracteres, espaços e símbolos especiais no lugar de número nos campos numéricos. Sua aplicação deve removê-los antes de passá-los para o MySQL ou sua aplicação deve gerar um erro. Passar valores não verificados ao MySQL é extramente perigoso!
  - Confira o tamanho dos dados antes de passá-los ao MySQL.
  - Considere ter sua aplicação conectando ao banco de dados utilizando um usuário diferente doq ue o que é utilizado com propósitos administrativos. Não forneça às suas aplicações mais privilégios de acesso do que elas necessitam.
- Usuários do PHP:
  - Confira a função `addslashes()`. No PHP 4.0.3, uma função `mysql_escape_string()` está disponível e é baseada na função com o mesmo nome da API C do MySQL.
- Usuários do API C do MySQL:
  - Confira a chamada API `mysql_escape_string()`.
- Usuários do MySQL:
  - Confira os modificadores `escape` e `quote` para consultas streams.
- Usuários do Perl DBI:
  - Confira o método `quote()` ou utilize aspas simples ou duplas.
- Usuários do Java JDBC:
  - Utilize um objeto `PreparedStatement` e aspas simples ou duplas.

- Não transmita dados sem criptografia na Internet. Estes dados são acessíveis para todos que tenham o tempo e habilidade para interceptá-lo e usá-lo para seu propósito próprio. No lugar, utilize um protocolo de criptografia como o SSL ou SSH. O MySQL suporta conexões SSL interno desde a versão 3.23.9. O repasse de portas do SSH pode ser usado para criar um tunel criptografado (e com compressão) para a comunicação.

- Aprenda a usar os utilitários `tcpdump` e `strings`. Para a maioria dos casos você pode conferir se o fluxo de dados do MySQL está ou não criptografado utilizando um comando parecido com este:

  ```
  shell> tcpdump -l -i eth0 -w - src or dst port 3306 | strings
  ```

  (Isto funciona sobre Linux e deve funcionar com pequenas modificações sob outros sistemas.) Alerta: Se você não ver dados não significa sempre que esteja criptografado. Se você necessita de alta segurança, você deve consultar um especialista em segurança.

4.3.2 Como Tornar o MySQL Seguro contra Crackers

Quando você conectar a um servidor MySQL, você normalmente deve usar uma senha. A senha não é transmitida em texto puro sobre a conexão, porém o algorítimo de criptografica não é muito forte e com algum esforço um atacante engenhoso pode quebrar a senha se ele conseguir capturar o tráfego entre o cliente e o servidor. Se a conexão entre o cliente e o servidor passar por uma rede não confiável, você deve usar um tunnel SSH para criptografar a comunicação.

Todas outras informações são transferidas como texto que podem ser lido por qualquer um que consiga ver a conexão. Se você se preocupa com isto, você pode usar o protocol de compressão (No MySQL versão 3.22 e superiores) para tornar o tráfico muito mais dificil de decifrar. Para deixar tudo ainda mais seguro você deve usar `ssh`. Você pode encontrar um cliente `ssh` open source em `http://www.openssh.org`, e um cliente `ssh` comercial em `http://www.ssh.com`. Com isto, você pode obter uma conexão TCP/IP critografada entre um servidor MySQL e um cliente MySQL.

Se você estiver usando o MySQL 4.0, você também pode usar o suporte interno OpenSSL Veja Seção 4.4.10 [Conexões seguras], Página 268.

Para deixar um sistema MySQL seguro, você deve considerar as seguintes sugestões:

- Utilize senhas para todos os usuários MySQL. Lembre-se que qualquer um pode logar como qualquer outra pessoa simplesmente com `mysql -u outro_usuário nome_bd` se `outro_usuário` não tiver senha. Isto é um procedimento comum com aplicações cliente/servidor que o cliente pode especificar qualquer nome de usuário. Você pode alterar a senha de todos seus usuários editando o script `mysql_install_db` antes de executá-lo ou somente a senha para o usuário `root` do MySQL desta forma:

  ```
  shell> mysql -u root mysql
  mysql> UPDATE user SET Password=PASSWORD('nova_senha')
      ->             WHERE user='root';
  mysql> FLUSH PRIVILEGES;
  ```

- Não execute o daemon do MySQL como o usuário `root` do Unix. Isto é muito perigoso, porque qualquer usuário com privilégios `FILE` estará apto a criar arquivos como o `root` (por exemplo, `~root/.bashrc`). Para prevenir esta situação, `mysqld` irá recusar a execução como `root` a menos que ele seja especificado diretamente usando a opção `--user=root`.

  O `mysqld` pode ser executado como um usuário normal sem privilégios. Você pode também criar um novo usuário Unix `mysql` para tornar tudo mais seguro. Se você executar o `mysqld` como outro usuário Unix, você não precisará alterar o usuário `root` na tabela `user`, porque nomes de usuário do MySQL não tem nada a ver com nomes

de usuários Unix. Para iniciar o `mysqld` como outro usuário Unix, adicione uma linha `user` que especifica o nome de usuário para o grupo `[mysqld]` do arquivo de opções '`/etc/my.cnf`' ou o arquivo de opções '`my.cnf`' no diretório de dados do servidor. Por exemplo:

```
[mysqld]
user=mysql
```

Estas opções configuram o servidor para iniciar como o usuário designado quando você o inicia manualmente ou usando `mysqld_safe` ou `mysql.server`. Para maiores detalhes, veja Seção A.3.2 [Changing MySQL user], Página 923.

- Não suportar links simbólicos para tabelas (Isto pode ser desabilitado com a opção `--skip-symlink`. Isto é muito importante caso você execute o `mysqld` como root, assim qualquer um que tenha acesso à escrita aos dados do diretório do mysqld podem apagar qualquer arquivo no sistema! Veja Seção 5.6.1.2 [Symbolic links to tables], Página 466.
- Verfique se o usuário Unix que executa o `mysqld` é o único usuário com privilégios de leitura/escrita nos diretórios de bancos de dados.
- Não forneça o privilégio **PROCESS** para todos os usuários. A saída de `mysqladmin processlits` mostra as consultas atualmente em execução, portanto qualquer usuário que consiga executar este comando deve ser apto a ver se outro usuário entra com uma consulta do tipo `UPDATE user SET password=PASSWORD('não_seguro')`.

  O `mysqld` reserva uma conexão extra para usuários que tenham o privilégio **process**, portanto o usuário `root` do MySQL pode logar e verificar a atividade do servidor mesmo se todas as conexões normais estiverem em uso.
- Não conceda o privilégio **FILE** a todos os usuários. Qualquer usuário que possua este privilégio pode gravar um arquivo em qualquer lugar no sistema de arquivos com os privilégios do daemon `mysqld`! Para tornar isto um pouco mais seguro, todos os arquivos gerados com `SELECT ... INTO OUTFILE` são lidos por todos, e não se pode sobrescrever arquivos existentes.

  O privilégio **FILE** pode também ser usado para ler qualquer arquivo acessível para o usuário Unix com o qual o servidor está sendo executado. Pode ocorrer abusos como, por exemplo, usar `LOAD DATA` para carregar o arquivo '`/etc/passwd`' em uma tabela, que pode então ser lido com `SELECT`.
- Se você não confia em seu DNS, você deve utilizar números IP no lugar de nomes de máquinas nas tabelas de permissão. De qualquer forma, você deve ter muito cuidado ao criar entradas de concessão utilizando valores de nomes de máquinas que contenham metacaractes!
- Se você deseja restrigir o número de conexões para um único usuário, você pode faze-lo configurando a variável `max_user_connections` no `mysqld`.

4.3.3 Opções de Inicialização para o `mysqld` em Relação a Segurança.

As seguintes opções do `mysqld` afetam a segurança:

`--local-infile[=(0|1)]`

Se alguém usa `--local-infile=0` então não de pode usar `LOAD DATA LOCAL INFILE`.

`--safe-show-database`
: Com esta opção, `SHOW DATABASES` retorna somente os bancos de dados nos quais o usuário tem algum tipo de privilégio. A partir da versão 4.0.2 esta opção está obsoleta e não faz nada (a opção está habilitada por padrão) já que agora temos o privilégio `SHOW DATABASES`. Veja Seção 4.4.1 [GRANT], Página 254.

`--safe-user-create`
: Se for habilitado, um usuário não consegue criar novos usuários com o comando `GRANT`, se o usuário não tiver privilégio de `INSERT` na tabela `mysql.user`. Se você desejar fornecer a um usuário acesso para só criar novos usuários com privilégios que o usuário tenha direito a conceder, você deve dar ao usuário o seguinte privilégio:

```
mysql> GRANT INSERT(user) ON mysql.user TO 'user'@'hostname';
```

Isto irá assegurar que o usuário não poderá alterar nenhuma coluna de privilégios diretamente, mas tem que usar o comando `GRANT` para conceder direitos para outros usuários.

`--skip-grant-tables`
: Esta opção desabilita no servidor o uso do sistema de privilégios. Isto dá a todos os usuários *acesso total* a todos os bancos de dados! (Você pode dizer a um servidor em execução para para uar as tabelas de permissões executando `mysqladmin flush-privileges` ou `mysqladmin reload`.)

`--skip-name-resolve`
: Nomes de máquinas não são resolvidos. Todos os valores da coluna `Host` nas tabelas de permissões devem ser números IP ou `localhost`.

`--skip-networking`
: Não permitir conexões TCP/IP sobre a rede. Todas as conexões para `mysqld` devem ser feitas via Sockets Unix. Esta opção não é possível em sistemas que usam MIT-pthreads, porque o pacote MIT-pthreads não suporta sockets Unix.

`--skip-show-database`
: Não permite o comando `SHOW DATABASES`, a menos que o usuário tenha o privilégio `SHOW DATABASES`. A partie da versão 4.0.2 você não deve mais precisar desta opção, já que o aceesso pode agora ser concedido especificamente com o privilégio `SHOW DATABASES`.

4.3.4 Detalhes de Segurança com `LOAD DATA LOCAL`

No MySQL 3.23.49 e MySQL 4.0.2 (4.0.13 no Windows), adicionamos algumas novas opções para lidar com possíveis detalhes de segurança junto ao `LOAD DATA LOCAL`.

Exstem dois problemas possíveis com o suporte a este comando:

Como a leitura deste arquivo é iniciada por um servidor, pode-se teoricamente criar um servidor MySQL corrigido que poderia ler qualquer arquivo na máquina cliente na qual o usuário atual tenha acesso, quando o cliente envia uma consulta a tabela.

Em um ambiente web onde os clientes estão conectados a um servidor web, um usuário poderia usar `LOAD DATA LOCAL` para ler qualquer arquivo no qual o processo do servidor web

tenha acesso de leitura (assumindo que um usuário poderia executar qualquer comando no servidor SQL).

Existem dois arquivos separados para isto:

Se você não configurar o MySQL com `--enable-local-infile`, então `LOAD DATA LOCAL` será disabilitado por todos os clientes, a menos que se chame `mysql_options(... MYSQL_OPT_LOCAL_INFILE, 0)` no cliente. Veja Seção 12.1.3.40 [`mysql_options()`], Página 809.

Para o cliente de linha de comando `mysql`, `LOAD DATA LOCAL` pode ser habilitado especificado a opção `--local-infile[=1]`, ou disabilitando com `--local-infile=0`.

Por padrão, todos os clientes e bibliotacas MySQL são compilados com `--enable-local-infile`, para ser compatível com o MySQL 3.23.48 e anterior.

Pode se desabilitar todos os comandos `LOAD DATA LOCAL` no servidor MySQL iniciando o `mysqld` com `--local-infile=0`.

No caso em que `LOAD DATA LOCAL INFILE` está disabilitado no servidor ou no cliente, você receberá a seguinte mensagem de erro (1148):

```
The used command is not allowed with this MySQL version
```

4.3.5 O Que o Sistema de Privilégios Faz

A função primária do sistema de privilégios do MySQL é autenticar um usuário a partir de uma determinada máquina e associar este usuário com privilégios a banco de dados como como **select**, **insert**, **update** e **delete**.

Funcionalidades adicionais incluem a habilidade de ter um usuário anônimo e conceder privilégio para funções específicas do MySQL como em `LOAD DATA INFILE` e operações administrativas.

4.3.6 Como o Sistema de Privilégios Funciona

O sistema de privilégios do MySQL garante que todos usuários possam fazer exatamente as operações que lhe é permitido. Quando você conecta a um servidor MySQL, sua identidade é determinada pela **maquina de onde você conectou** e **o nome de usuário que você especificou**. O sistema concede privilégios de acordo com sua identidade e **com o que você deseja fazer**.

O MySQL considera tanto os nomes de máquinas como os nomes de usuários porque existem poucas razões para assumir que um determinado nome de usuário pertence a mesma pessoa em todo lugar na Internet. Por exemplo, o usuário `bill` que conecta de `whitehouse.gov` não deve necessariamente ser a mesma pessoa que o usuário `bill` que conecta da `microsoft.com` O MySQL lida com isto, permitindo a distinção de usuários em diferentes máquinas que podem ter o mesmo nome: Você pode conceder a `bill` um conjunto de privilégios para conexões de `whitehouse.gov` e um conjunto diferente de privilégios para conexões de `microsoft.com`.

O controle de acesso do MySQL é composto de dois estágios:

- Estágio 1: O servidor confere se você pode ter acesso ou não.
- Estágio 2: Assumindo que você pode conectar, o servidor verifica cada requisição feita para saber se você tem ou não privilégios suficientes para realizar a operação. Por exemplo, se você tentar selecionar linha de uma tabela em um banco de dados ou

apagar uma tabela do banco de dados, o servidor se certifica que você tem o privilégio **select** para a tabela ou o privilégio **drop** para o banco de dados.

Note que se os seus privilégios são alterados (tanto por você quanto por outro) enquanto você está conectado, estas alterações não irão necessariamente ter efeito com a sus próxima consulta ou consultas. Veja Seção 4.4.3 [Privilege changes], Página 260 para maiores detalhes.

O servidor utiliza as tabelas `user`, `db` e `host` no banco de dados `mysql` em ambos estágios do controle de acesso. Os campos nestas tabelas de permissão são detalhados abaixo:

Nome da Tabela	**user**	**db**	**host**
Campos de Escopo	`Host`	`Host`	`Host`
	`User`	`Db`	`Db`
	`Password`	`User`	
Campos de Privilégio	`Select_priv`	`Select_priv`	`Select_priv`
	`Insert_priv`	`Insert_priv`	`Insert_priv`
	`Update_priv`	`Update_priv`	`Update_priv`
	`Delete_priv`	`Delete_priv`	`Delete_priv`
	`Index_priv`	`Index_priv`	`Index_priv`
	`Alter_priv`	`Alter_priv`	`Alter_priv`
	`Create_priv`	`Create_priv`	`Create_priv`
	`Drop_priv`	`Drop_priv`	`Drop_priv`
	`Grant_priv`	`Grant_priv`	`Grant_priv`
	`References_priv`	`References_priv`	`References_priv`
	`Reload_priv`		
	`Shutdown_priv`		
	`Process_priv`		
	`File_priv`		
	`Show_db_priv`		
	`Super_priv`		
	`Create_tmp_table_priv`	`Create_tmp_table_priv`	`Create_tmp_table_priv`
	`Lock_tables_priv`	`Lock_tables_priv`	`Lock_tables_priv`
	`Execute_priv`		
	`Repl_slave_priv`		
	`Repl_client_priv`		
	`ssl_type`		
	`ssl_cypher`		
	`x509_issuer`		
	`x509_cubject`		
	`max_questions`		

	max_updates	
	max_	
	connections	

No segundo estágio do controle de acesso (verificação da solicitação), o servidor pode, se a solicitação involver tabelas, consultar adicionalmente as tabelas `tables_priv` e `columns_priv`. Os campos nestas tabelas são mostrados abaixo:

Nome da tabela	`tables_priv`	`columns_priv`
Campos de escopop	`Host`	`Host`
	`Db`	`Db`
	`User`	`User`
	`Table_name`	`Table_name`
		`Column_name`
Campos de privilégio	`Table_priv`	`Column_priv`
	`Column_priv`	
Outros campos	`Timestamp`	`Timestamp`
	`Grantor`	

Cada tabela de permissões contêm campos de escopo e campos de privilégios.

Campos de escopo determinam o escopo de cada entrada nas tabelas, isto é, o contexto no qual a entrada se aplica. Por exemplo, uma entrada na tabela `user` com valores `Host` e `User` de `'thomas.loc.gov'` e `'bob'` devem ser usados para autenticar conexões feitas ao servidor por `bob` da máquina `thomas.loc.gov`. De maneira similar, uma entrada na tabela `db` com campos `Host`, `User` e `Db` de `'thomas.loc.gov'`, `'bob'` e `'reports'` devem ser usados quando `bob` conecta da máquina `thomas.loc.gov` para acessar o banco de dados `reports`. As tabelas `tables_priv` e `columns_priv` contem campos de escopo indicando as combinações de tabelas ou tabela/coluna para o qual cada entrada se aplica.

Para propósitos de verificação de acessos, comparações de valores `Host` são caso insensitivo, valores `User`, `Password`, `Db` e `Table_name` são caso sensitivo. Valores `Column_name` são caso insensitivo no MySQL versão 3.22.12 ou posterior.

Campos de privilégios indicam os privilégios concedidos por uma entrada na tabela, isto é, quais operações podem ser realizadas. O servidor combina as informações de várias tabelas de concessão para formar uma descrição completa dos privilégios de um usuário. As regras usadas para fazer isto são descritas em Seção 4.3.10 [Request access], Página 242.

Campos de escopo são strings, declaradas como mostrado abaixo; os valores padrão para cada é a string vazia:

Nome do Campo	**Tipo**	
`Host`	`CHAR(60)`	
`User`	`CHAR(16)`	
`Password`	`CHAR(16)`	
`Db`	`CHAR(64)`	(`CHAR(60)` para as tabelas `tables_priv` e `columns_priv`)
`Table_name`	`CHAR(60)`	
`Column_name`	`CHAR(60)`	

Nas tabelas `user`, `db` e `host`, todos campos de privilégios são declarados como `ENUM('N','Y')` — cada um pode ter um valor de `'N'` ou `'Y'` e o valor padrão é `'N'`.

Nas tabelas `tables_` e `columns_priv`, os campos de privilégios são declarados como campos `SET`:

Nome de tabela	Nome do campo	Possíveis elementos do conjunto
tables_priv	Table_ priv	'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter'
tables_priv	Column_ priv	'Select', 'Insert', 'Update', 'References'
columns_ priv	Column_ priv	'Select', 'Insert', 'Update', 'References'

De maneira resumida, o servidor utiliza as tabelas de permissões desta forma:

- Os campos de escopo da tabela `user` determinam quando permitir ou aceitar conexões. Para conexões permitidas, qualquer privilégio concedido em uma tabela `user` indica o privilégio global (superusuário) do usuário. Estes privilégios se aplicam a **todos** os bancos de dados no servidor.
- As tabelas `db` e `host` são usadas juntas:
  - Os campos de escopo da tabela `db` determinam quais usuários podem acessar determinados bancos de dados de máquinas determinadas. Os campos de privilégios determinam quais operações são permitidas.
  - A tabela `host` é usada como uma extensão da tabela `db` quando você quer que uma certa entrada na tabela `db` seja aplicada a diversas máquinas. Por exemplo, se você deseja que um usuário esteja apto a usar um banco de dados a partir de diversas máquinas em sua rede, deixe o campo `Host` vazio no registro da tabela `db`, então popule a tabela `Host` com uma entrada para cada uma das máquinas. Este mecanismo é descrito com mais detalhes em Seção 4.3.10 [Request access], Página 242.
- As tabelas `tables_priv` e `columns_priv` são similares à tabela `db`, porém são mais finas: Elas se aplicam ao nível de tabelas e colunas em vez do nível dos bancos de dados.

Perceba que os privilégios administrativos (`RELOAD`, `SHUTDOWN` e etc) são especificados somente na tabela `user`. Isto ocorre porque operações administrativas são operações no próprio servidor e não são específicas e não específicas dos bancos de dados, portanto não existe razão para listar tais privilégios nas outras tabelas de permissão. De fato, somente a tabela `user` necessita ser consultada para determinar se você pode ou não realizar uma operação administrativa.

O privilégio **FILE** também só é especificado na tabela `user`. Ele não é um privilégio administrativo, mas sua habilidade para ler ou escrever arquivo no servidor é independtende do banco de dados que você está acessando.

O servidor `mysqld` le o conteúdo das tabelas de permissões uma vez, quando é iniciado. Alterações nas tabelas de permissões tem efeito como indicado em Seção 4.4.3 [Alterçao de privilégios], Página 260.

Quando você modifica o conteúdo das tabelas de permissões, é uma boa idéia ter certeza que suas alterações configuraram os privilégios da forma desejada. Para ajuda no diagnostico de problemas, veja Seção 4.3.12 [Access denied], Página 250. Para conselhos sobre asssuntos de segurança, Veja Seção 4.3.2 [Security], Página 229.

Uma ferramenta de diagnóstico útil é o script `mysqlaccess`, que Yves Carlier fornece na distribuição MySQL. Chame `mysqlaccess` com a opção `--help` para descobrir como ele funciona. Perceba que o `mysqlaccess` confere o acesso usando somente as tabelas `user`, `db` e `host`. Ele não confere privilégios no nível de tabelas ou colunas.

4.3.7 Privilégios Fornecidos pelo MySQL

Informações sobre privilégios de usuários são armazenados nas tabelas `user`, `db`, `host`, `tables_priv` e `columns_priv` no banco de dados chamado `mysql`. O servidor MySQL lê o conteúdo destas tabelas quando ele inicia e sob as circunstâncias indicadas em Seção 4.4.3 [Privilege changes], Página 260.

Os nomes usados neste manual que se referem-se aos privilégios fornecidos pelo MySQL são vistos abaixo juntos com o nome da coluna associada com cada privilégio nas tabelas de permissão e o contexto em que o privilégio se aplica. Informações adicionais sobre o significado de cada privilégio pode ser encontrado em Seção 4.4.1 [`GRANT`], Página 254.

Privilégio	**Coluna**	**Contexto**
`ALTER`	`Alter_priv`	tabelas
`DELETE`	`Delete_priv`	tabelas
`INDEX`	`Index_priv`	tabelas
`INSERT`	`Insert_priv`	tabelas
`SELECT`	`Select_priv`	tabelas
`UPDATE`	`Update_priv`	tabelas
`CREATE`	`Create_priv`	banco de dados, tabelas, ou índices
`DROP`	`Drop_priv`	banco de dados ou tabelas
`GRANT`	`Grant_priv`	banco de dados ou tabelas
`REFERENCES`	`References_priv`	banco de dados ou tabelas
`CREATE TEMPORARY TABLES`	`Create_tmp_tabela_priv`	administração do servidor
`EXECUTE`	`Execute_priv`	administração do servidor
`FILE`	`File_priv`	acessa a arquivos no servidor
`LOCK TABLES`	`Lock_tabelas_priv`	administração do servidor
`PROCESS`	`Process_priv`	administração do servidor
`RELOAD`	`Reload_priv`	administração do servidor
`REPLICATION CLIENT`	`Repl_client_priv`	administração do servidor
`REPLICATION SLAVE`	`Repl_slave_priv`	administração do servidor
`SHOW DATABASES`	`Show_db_priv`	administração do servidor

SHUTDOWN	Shutdown_ priv	administração do servidor
SUPER	Super_priv	administração do servidor

Os priviláegios SELECT, INSERT, UPDATE e DELETE permitem realizar operações em registros nas tabelas existentes em um banco de dados.

Instruções SELECT necessitam do privilégio **select** somente se ele precisar recuperar registros de uma tabela. Você pode executar certas instruções SELECT mesmo sem permissão para acessar algum dos bancos de dados no servidor. Por exemplo, você pode usar o cliente mysql como uma simples calculadora:

```
mysql> SELECT 1+1;
mysql> SELECT PI()*2;
```

O privilégio INDEX permite a criação ou remoção de índices.

O privilégio ALTER permite utilizar ALTER TABLE.

Os privilégios CREATE e DROP permitem a criação de novos bancos de dados e tabelas, ou a remoção de bancos de dados e tabelas existentes.

Perceba que se for concedido o privilégio DROP no banco de dados mysql para algum usuário, este usuário pode remover o banco de dados no qual os privilégios de acesso do MySQL estão armazenados!

O privilégio GRANT permite a você fornecer a outros usuários os privilégios que você mesmo possui.

O privilégio FILE fornece permissão para ler e escrever arquivos no servidor usando instruções LOAD DATA INFILE e SELECT ... INTO OUTFILE. Qualquer usuário que tenha este privilégio pode ler ou gravar qualquer arquivo que o servidor MySQL possa ler ou escrever. O usuário também pode ler qualquer arquivo no diretório de banco de dados atual. O usuário não pode, no entanto, alterar qualquer arquivo existente.

Os privilégios restantes são usados para operações administrativas, que são realizadas utilizando o programa mysqladmin. A tabela abaixo mostra quais comandos do mysqladmin cada privilégio administrativos permite a execução:

Privilégio	**Comandos permitidos**
RELOAD	reload, refresh, flush-privileges, flush-hosts, flush-logs, and flush-tables
SHUTDOWN	shutdown
PROCESS	processlist
SUPER	kill

O comando reload diz ao servidor para recarregar as tabelas de permissões. O comando refresh descarrega todas as tabelas e abre e fecha os arquivos de log. flush-privileges é um sinônimo para reload. Os outros comandos flush-* realizam funções similares ao refresh mas são mais limitados no escopo e podem ser preferíveis em alguns casos. Por exemplo, se você deseja descarregar apenas os arquivos log, flush-logs é uma melhor escolha do que refresh.

O comando shutdown desliga o servidor.

O comando processlist mostra informações sobre as threads em execução no servidor. O comando kill mata threads no servidor. Você sempre poderá mostrar ou matar suas

próprias threads, mas você precisa do privilégio `PROCESS` para mostrar e privilégio `SUPER` para matar threads iniciadas por outros usuários. Veja Seção 4.6.7 [KILL], Página 301.

É uma boa idéia em geral conceder privilégios somente para aqueles usuários que necessitem deles, mas você deve ter muito cuidado ao conceder certos privilégios:

- O privilégio **grant** permite aos usuários repassarem seus privilégios a outros usuários. Dois usuários com diferentes privilégios e com o privilégio **grant** conseguem combinar seus privilégios.
- O privilégio **alter** pode ser usado para subverter o sistema de privilégios renomeando as tabelas.
- O privilégio `FILE` pode ser usado com abuso para ler qualquer arquivo de leitura no servidor em uma tabela de banco de dados, o conteúdo pode ser acessando utilizando `SELECT`. Isto inclui o conteúdo de todos os bancos de dados hospedados pelo servidor!
- O privilégio `SHUTDOWN` pode ser utilizado para negar inteiramente serviços para oturos usuários, terminando o servidor.
- O privilégio `PROCESS` pode ser usado para ver o texto das consultas atualmente em execução, incluindo as consultas que configuram ou alteram senhas.
- Privilégios no banco de dados `mysql` pode ser utilizado para alterar senhas e outras informações de privilégio de acesso. (Senhas são armazenadas criptografadas, portanto um usuário malicioso não pode simplesmente lê-las para saber as senhas em texto puro). Se fosse possível acessar a coluna password do banco `mysql.user`, seria possível logar ao servidor MySQL como outro usuário. (Com privilégios suficientes, o mesmo usuário pode trocar a senha por outra diferente.)

Existema algumas coisas que você não pode fazer com o sistem de privilégios do MySQL:

- Você não pode especificar explicitamente que um determinado usuário deve ter acesso negado. Você não pode explicitamente comparar um usuário e depois recusar sua conexão.
- Você não pode especificar que um usuário tenha privilégios para criar ou remover tabelas em um banco de dados, mas não possa criar ou remover o banco de dados.

4.3.8 Conectando ao Servidor MySQL

Programas clientes do MySQL geralmente necessitam de parâmetros de conexão quando você precisar acessar um servidor MySQL: a máquina na qual você deseja se conectar, seu nome de usuário e sua senha. Por exemplo, o cliente `mysql` pode ser iniciado desta forma (argumentos opcionais são colocandos entre '`[`' e '`]`'):

```
shell> mysql [-h nome_máquina] [-u nome_usuário] [-psua_senha]
```

Formas alternativas das opções `-h`, `-u` e `-p` são `--host=nome_máquina`, `--user=nome_usuário` e `--password=sua_senha`. Perceba que não existe *espaço* entre `-p` ou `--password=` e a senha que deve vir a seguir.

NOTA: Especificar a senha na linha de comando não é seguro! Qualquer usuário no seus sistema pode saber sua senha digitando um comando do tipo: `ps auxww`. Veja Seção 4.1.2 [Option files], Página 216.

O `mysql` utiliza valores padrão para parâmetros de conexão que não são passados pela linha de comando:

- O nome padrão da máquina (hostname) é `localhost`.
- O nome de usuário padrão é o mesmo nome do seu usuário no Unix.
- Nenhuma senha é fornecida se faltar o parâmetro `-p`.

Então, para um usuário Unix `joe`, os seguintes comandos são equivalentes:

```
shell> mysql -h localhost -u joe
shell> mysql -h localhost
shell> mysql -u joe
shell> mysql
```

Outros clientes MySQL comportam-se de forma similar.

Em sistemas Unix, você pode especificar valores padrões diferentes para serem usados quendo você faz uma conexão, assim você não precisa digitá-los na linha de comando sempre que chamar o programa cliente. Isto pode ser feito de várias maneiras:

- Podem ser especificados parâmetros de conexão na seção `[client]` do arquivo de configuração '`.my.cnf`' no seu diretório home. A seção relevante do arquivo deve se parecer com isto:

  ```
  [client]
  host=nome_máquina
  user=nome_usuário
  password=senha_usuário
  ```

  Veja Seção 4.1.2 [Option files], Página 216.
- Você pode especificar parâmetros de conexão utilizando variáveis de ambiente. O nome de máquina pode ser especificado para o `mysql` utilizando a variável `MYSQL_HOST`. O nome do usuário MySQL pode ser especificado utilizando `USER` (isto é somente para Windows). A senha pode ser especificada utilizando `MYSQL_PWD` (mas isto não é seguro; veja a próxima seção). Veja Apêndice F [Variáveis de ambiente], Página 1092.

4.3.9 Controle de Acesso, Estágio 1: Verificação da Conexão

Quando você tenta se conectar a um servidor MySQL, o servidor aceita ou rejeita a conexão baseado na sua identidade e se pode ou não verificar sua identidade fornecendo a senha correta. Senão, o servidor nega o acesso a você completamente. De outra forma, o servidor aceita a conexão, entra no estágio 2 e espera por requisiçiões.

Sua identidade é baseada em duas partes de informação:

- A máquina de onde está conectando
- Seu nome de usuário no MySQL

A conferência da identidade é feita utilizando os tres campos de escopo da tabela `user` (`Host`, `User` e `Password`). O servidor aceita a conexão somente se uma entrada na tabela `user` coincidir com a máquina, nome de usuário e a senha fornecidos.

Valores dos campos escopo na tabela `user` podem ser especificados como segue:

- Um valor `Host` deve ser um nome de máquina ou um número IP ou `'localhost'` para indicar a máquina local.
- Você pode utilizar os metacaracteres '`%`' e '`_`' no campo `Host`.

- Um valor `Host` de `'%'` coincide com qualquer nome de máquina.
- Um valor `Host` em branco significa que o privilégio deve ser adicionado com a entrada na tabela `host` que coincide com o nome de máquina fornecido. Você pode encontrar mais informações sobre isto no próximo capítulo.
- Como no MySQL Versão 3.23, para valores `Host` especificados como números IP, você pode especificar uma máscara de rede indicando quantos bits de endereço serão usados para o número da rede. Por exemplo:

```
mysql> GRANT ALL PRIVILEGES ON db.*
    -> TO david@'192.58.197.0/255.255.255.0';
```

  Isto permitirá que todos a se conectarem a partir de determinado IP cuja condição seguinte seja verdadeira:

```
IP_usuário & máscara_rede = ip_maquina.
```

  No exemplo acima todos IPs no Intervalo 192.58.197.0 - 192.58.197.255 podem se conectar ao servidor MySQL.
- Metacaracteres não são permitidos no campo `User`, mas você pode especificar um valor em branco, que combina com qualquer nome. Se a entrada na tabela `user` que casa com uma nova conexão tem o nome do usuário em branco, o usuário é considerado como um usuário anônimo (o usuário sem nome), em vez do nome que o cliente especificou. Isto significa que um nome de usuário em branco é usado para todos as verificações de acessos durante a conexão. (Isto é, durante o estágio 2).
- O campo `Password` pode ficar em branco. O que não significa que qualquer senha possa ser usada, significa que o usuário deve conectar sem especificar uma senha.

Valores de `Password` que não estão em branco são apresentados como senhas criptografadas. O MySQL não armazena senhas na forma de texto puro para qualquer um ver. Em vez disso, a senha fornecida por um usuário que está tentando se conectar é criptografada (utilizando a função `PASSWORD()`). A senha criptografada é então usada quando o cliente/servidor estiver conferindo se a senha é correta (Isto é feito sem a senha criptografada sempre trafegando sobre a conexão.) Perceba que do ponto de vista do MySQL a senha criptografada é a senha REAL, portanto você não deve passá-la para ninguém! Em particular, não forneça a usuários normais acesso de leitura para as tabelas no banco de dados `mysql`! A partir da versão 4.1, o MySQL emprega um mecanismo de senha e login diferente que é seguro mesmo se fizerem um sniff nos pacotes TCP/IP e/ou o banco de dados mysql é capturado.

Os exemplos abaixo mostram várias combinações de valores de `Host` e `User` nos registros da tabela `user` aplicando a novas conexões:

Valor em `host`	**Valor em** `user`	**Conexões casadas com o registro**
`'thomas.loc.gov'`	`''`	Qualquer usuário, conectando de `thomas.loc.gov`
`'%'`	`'fred'`	`fred`, conectando a partir de qualquer máquina
`'%'`	`''`	Qualquer usuário, conectando a partir de qualquer máquina
`'%.loc.gov'`	`'fred'`	`fred`, conectando de qualquer máquina do domínio `loc.gov`

`'x.y.%'`	`'fred'`	`fred`, conectando de `x.y.net`, `x.y.com`,`x.y.edu`, etc. (Isto provavelmente não é útil)
`'144.155.166.177'`	`'fred'`	`fred`, conectando da máquina com endereço IP `144.155.166.177`
`'144.155.166.%'`	`'fred'`	`fred`, conectando de qualquer máquina na subrede de classe C `144.155.166`
`'144.155.166.0/255.255.255.0'`	`'fred'`	o mesmo que no exemplo anterior

Como você pode usar valores coringas de IP no campo `Host` (por exemplo, `'144.155.166.%'` combina com todas máquinas em uma subrede), existe a possibilidade que alguém possa tentar explorar esta capacidade nomeando a máquina como `144.155.166.algumlugar.com`. Para evitar tais tentativas, O MySQL desabilita a combinação com nomes de máquina que iniciam com dígitos e um ponto. Portanto se você possui uma máquina nomeada como `1.2.foo.com`, este nome nunca irá combinar com uma coluna `Host` das tabelas de permissões. Somente um número IP pode combinar com um valor coringa de IP.

Uma conexão de entrada pode coincidir com mais de uma entrada na tabela `user`. Por exemplo, uma conexão a partir de `thomas.loc.gov` pelo usuário `fred` pode combinar com diversas das entradas vistas na tabela anterior. Como o servidor escolhe qual entrada usar se mais de uma coincide? O servidor resolve esta questão ordenando a tabela `user` no tempo de inicialização, depois procura pelas entradas na ordem da classificação quando um usuário tenta se conectar. A primeira entrada que coincidir é a que será usada.

A ordenação da tabela `user` funciona da forma mostrada a seguir. Suponha que a tabela `user` se pareça com isto:

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| %         | root     | ...
| %         | jeffrey  | ...
| localhost | root     | ...
| localhost |          | ...
+-----------+----------+-
```

Quando o servidor lê a tabela, ele ordena as entradas com os valores mais específicos de `Host` primeiro (`'%'` na coluna `Host` significa "qualquer máquina" e é menos específico). Entradas com o mesmo valor `Host` são ordenadas com os valores mais específicos de `User` primeiro (um valor em branco na coluna `User` significa "qualquer usuário" e é menos específico). O resultado da tabela `user` ordenada ficaria assim:

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| localhost | root     | ...
| localhost |          | ...
| %         | jeffrey  | ...
| %         | root     | ...
+-----------+----------+-
```

Quando uma conexão é iniciada, o servidor procura entre as entradas ordenadas e utiliza a primeira entrada coincidente. Para uma conexão a partir de `localhost` feito por `jeffrey`,

as entradas com `'localhost'` na coluna `Host` coincide primeiro. Destas, a entrada com o nome do usuário em branco combina com o nome da máquina e o nome do usuário. (A entrada `'%'/'jeffrey'` também casaria, mas ela não é a primeira entrada coincidente na tabela.

Aqui está outro exemplo. Suponha que a tabela `user` fosse assim:

```
+----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
| %              | jeffrey  | ...
| thomas.loc.gov |          | ...
+----------------+----------+-
```

A tabela ordenada pareceria com isto:

```
+----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
| thomas.loc.gov |          | ...
| %              | jeffrey  | ...
+----------------+----------+-
```

Uma conexão a partir de `thomas.loc.gov` feita por `jeffrey` coincide com a primeira entrada, no entanto, uma conexão de `whitehouse.gov` fetia por `jeffrey` coincidiria com a segunda entrada na tabela.

Um erro comum é pensar que para um determinado usuário, todas as entradas que citam explicitamente este usuário serão usadas primeiro quando o usuário tentar encontrar uma combinação para a conexão. Simplesmente isto não é verdade. O exemplo anterior ilustra isto, onde uma conexão de `thomas.loc.gov` feita por `jeffrey` combina primeiro não com a entrada contendo `'jeffrey'` no valor do campo `user`, mas sim pela entrada sem o nome de usuário!

Se você tiver problemas conectando ao servidor, imprima a tabela `user` e ordene-a na manualmente para ver onde se deu o primeiro coincidência de valores. Se a conexão obtiver sucesso mas os seus privilégios não são os esperados, você pode usar a função `CURRENT_USER()` (nova na versão 4.0.6) para ver com qual combinação usuário/máquina a sua conexão coincide. Veja Seção 6.3.6.2 [`CURRENT_USER()`], Página 547.

4.3.10 Controle de Acesso, Estágio 2: Verificação da Requisição

Uma vez estabelecida uma conexão, o servidor entra no 2o estágio. Para cada requisição que vem na conexão, o servidor verifica se você tem privilégios suficientes para realizá-la, baseado nas operações que você deseja fazer. É aqui que os campos de concessões nas tabelas de permissões entram em ação. Estes privilégios pode vir de qualquer uma das tabelas `user`, `db`, `host`, `tables_priv` ou `columns_priv`. As tabelas de permissões são manipuladas com os comandos `GRANT` e `REVOKE`. Veja Seção 4.4.1 [`GRANT`], Página 254. (Você pode achar útil fazer referencia a Seção 4.3.6 [Privileges], Página 232, que lista os campos presentes em cada uma das tabelas de permissões.)

A tabela `user` concede privilégios que são especificados por você em uma base global e que se aplicam sem importar qual é o banco de dados atual. Por exemplo, se a tabela `user`

concede a alguém o privilégio **delete**, este usuário pode apagar linhas de qualquer banco de dados no servidor! Em outras palavras, privilégios na tabela `user` são privilégios de superusuário. O correto é conceder privilégios na tabela `user` apenas para superusuários tais como os administradores de servidor ou de bancos de dados. Para outros usuários, você deve deixar os privilégios na tabela `user` configurados para `'N'` e conceder privilégios somente em bancos de dados específicos, utilizando as tabelas `db` e `host`.

As tabelas `db` e `host` concedem privilégios para bancos de dados específicos. Valores nos campos de escopo podem ser especificados como a seguir:

- Os metacaracteres '`%`' e '`_`' podem ser usados nos campos `Host` e `Db` de ambas tabelas. Se você deseja usar um caracter '`_`' como parte de um nome de banco de dados, especifique-o como '`\_`' no comando `GRANT`.
- O valor `'%'` em `Host` na tabela `db` significa "qualquer máquina." Um valor em branco em `Host` na tabela `db` significa "consulte a tabela `host` para informação adicional."
- O valor `'%'` ou em branco no campo `Host` na tabela `host` significa "qualquer máquina."
- O valor `'%'` ou em branco no campo `Db` de ambas as tabelas significa "qualquer banco de dados."
- O valor em branco no campo `User` em ambas tabelas coincide com o usuário anônimo.

As tabelas `db` e `host` são lidas e ordenadas quando o servidor inicia (ao mesmo tempo que ele lê a tabela `user`). A tabela `db` é ordenada nos campos de escopo `Host`, `Db` e `User` e a tabela `host` é ordenada nos campos de escopo `Host` e `Db`. Assim como na tabela `user`, a ordenação coloca os valores mais específicos no início e os menos específicos por último, e quando o servidor procura por entradas coincidentes, ele usa a primeira combinação que encontrar.

As tabelas `tables_priv` e `columns_priv` concedem privilégios específicos para tabelas e campos. Valores nos campos escopo podem ser especificados como a seguir:

- Os meta caracteres '`%`' e '`_`' podem ser usados no campo `Host` de ambas tabelas.
- O valor `'%'` ou em branco no campo `Host` em ambas tabelas significam "qualquer máquina"
- Os campos `Db`, `Table_name` e `Column_name` não podem conter meta caracteres ou serem brancos em ambas tabelas.

As tabelas `tables_priv` e `columns_priv` são ordenadas nos campos `Host`, `DB` e `User`. Isto é parecido com a ordenação da tabela `db`, no entanto, a ordenação é mais simples porque somente o campo `Host` pode conter meta caracteres.

O processo de verificação da requisição é descrito abaixo. (Se você já está familiarizado com o código de verificação de acesso, você irá perceber que a descrição aqui é um pouco diferente do algorítimo usado no código. A descrição é equivalente ao que o código realmente faz; ele só é diferente para tornar a explicação mais simples.)

Para requisições administrativas (`SHUTDOWN`, `RELOAD`, etc.), o servidor confere somente a entrada da tabela `user`, porque ela é a única tabela que especifica privilégios administrativos. O acesso é concedido se o registro permitir a operação requisitada ou negado caso o contrário. Por exemplo, se você deseja executar `mysqladmin shutdown` mas a entrada em sua tabela `user` não lhe concede o privilégio `SHUTDOWN`, o acesso é negado mesmo sem consultar as tabelas `db` ou `host`. (elas não contém o campo `Shutdown_priv`, portanto não existe esta necessidade.)

Para requisições relacionadas aos bancos de dados (`insert`, `udpdate`, etc.), o servidor primeiro confere os privilégios globais do usuário consultando as entradas da tabela `user`. Se a entrada permitir a operação requisitada, o acesso é concedido. Se os privilégios globais na tabela `user` são insuficientes, o servidor determina os privilégios específicos de banco de dados para o usuário consultando as tabelas `db` e `host`:

1. O servidor consulta a tabela `db` por uma combinação nos campos `Host`, `Db` e `User`. Os campos `Host` e `User` são comparados com o nome da máquina e o nome do usuário que faz a requisição. O campo `Db` é comparado com o banco de dados que o usuário deseja acessar. Se não existir entradas coincidentes para o `Host` e `User`, o acesso é negado.
2. Se existir uma combincação com a entrada da tabela `db` e seu campo `Host` não estiver em branco, aquela entrada define os privilégios especificos do banco de dados do usuario.
3. Se o registro coincidente da tabela `db` tiver o campo `Host` em branco, significa que a tabela `host` enumera quais máquinas são permitidas acessar o banco de dados. Neste caso, uma consulta adicional é feita na tabela `host` para encontrar uma valores coincidentes nos campos `Host` e `Db`. Se nenhuma entrada na tabela `host` coincide, o acesso é negado. Se existir uma coincidência, os privilégios específicos de bancos de dados para o usuário são computados como a interseção (*não* a união!) dos privilégios nas entradas das tabelas `db` e `host`, isto é, os privilégios que são `'Y'` em ambas entradas. (Desta forma você pode conceder privilégios gerais em entradas na tabela `db` e então restringi-los em uma base de máquina a máquina utilizando as entradas da tabela `host`.)

Depois de determinar os privilégios específicos do banco de dados concedido pelas entradas nas tabelas `db` e `host`, o servidor os adiciona aos privilégios globais concedidos pela tabela `user`. Se o resultado permitir a operação requisitada, o acesso será concedido. De outra forma, o servidor consulta os privilégios de tabelas e campos do usuario nas tabelas `tables_priv` e `columns_priv` e os adiciona aos privilégios do usuário. O acesso será permitido ou negado baseado no resultado.

Expresso em termos booleanos, a descrição precedente de como os privilégios de um usuário são calculados podem ser resumido assim:

```
global privileges
OR (database privileges AND host privileges)
OR table privileges
OR column privileges
```

Ele pode não ser aparente porque, se os privilégios da entrada global de `user` são inicialmente insuficientes para a operação requisitada, o servidor adiciona estes privilégios mais tarde aos privilégios específicos de banco de dados, tabelas e colunas. A razão é que uma requisição pode exigir mais que um tipo de privilégio. Por exemplo, se você executar uma instrução `INSERT ... SELECT`, você precisa dos privilégios `INSERT` e `SELECT`. Seu privilégio pode ser tal que a entrada da tabela `user` concede um privilégio e a entrada da tabela `db` concede o outro. Neste caso, você tem os privilégios necessários para realizar a requisição, mas o servidor não pode obtê-los de ambas as tabelas por si próprio; os privilégios concedidos pelas entradas em ambas as tabelas de ser combinados.

A tabela `host` pode ser usada para manter uma lista dos servidores seguros.

Na Tcx, a tabela `host` contém uma lista de todas as máquina na rede local. A elas são concedidos todos os privilégios.

Você pode também usar a tabela `host` para indicar máquinas que *não* são seguras. Suponha que você tenha uma máquina `public.your.domain` que está localizada em uma área pública que você não considera segura. Você pode permitir o acesso a todas as máquinas de sua rede exceto a esta máquina usando entradas na tabela `host` desta forma:

```
+--------------------+----+-
| Host               | Db | ...
+--------------------+----+-
| public.your.domain | %  | ... (todos os privilégios configurados para 'N')
| %.your.domain      | %  | ... (todos os privilégios configurados para 'Y')
+--------------------+----+-
```

Naturalmente, você deve sempre testar suas entradas nas tabelas de permissões (por exemplo, usar o `mysqlaccess` para ter certeza que os privilégios de acesso estão atualmente configurados da forma que você imagina.

4.3.11 Hashing de Senhas no MySQL 4.1

As contas de usuários do MySQL estão lisatadas na tabela `user` do banco de dados `mysql`. Para cada conta do MySQL é definida uma senha, no entanto o que está armazenado na coluna `Password` da tabela `user` não seja uma versão da senha em texto puro, mas um valor hash computado para ela. Valores hash de senha são calculados pela função `PASSWORD()`.

O MySQL usa senhas em duas fases da comunicação cliente/servidor:

- Primeiro, quando um cliente tenta se conectar ao servidor, existe uma etapa de autenticação inicial na qual o cliente deve apresentar uma senha que combina com o valor hash armazenado na tabela de usuários para a conta que aquele cliente deseja usar.
- Em segundo lugar, depois que o cliente conecta, ele pode configurar ou alterar o hash da senha para as contas listadas na tabela de usuário (se ele tiver privilégios suficientes).O cliente pode fazer isto usando a função PASSWORD() para gerar uma hash da senha ou usando as instruções GRANT ou SET PASSWORD.

Em outra palavras, o servidor *usa* valores hash durante a autenticação quando um cliente tenta a primeira conexão. O servidor *gera* os valores hash se um cliente conectado chama a função `PASSWORD()` ou usa uma instrução `GRANT` ou `SET PASSWORD` para definir ou alterar uma senha.

O mecanismo de hash da senha foi atualizado no MySQL 4.1 para fornecer melhor segurança e reduzir os riscos de senhas serem roubadas. No entanto, Este novo mecanismo só é interpretado pelo servidor 4.1 e clientes 4.1, que podem resultar em alguns problemas de compatibilidade. Um cliente 4.1 pode conectar a um servidor pre-4.1, porque o cliente entende tanto o antigo quanto o novo mecanismo hash de senha. No entanto, um cliente pre-4.1 que tentar se conectar a um servidor 4.1 pode encontrar dificuldades. Por exemplo, um cliente `mysql` 4.0 que tentar se conectar a um servidor 4.1 pode falhar com a seguinte mensagem de erro:

```
shell> mysql
Client does not support authentication protocol requested
by server; consider upgrading MySQL client
```

A seguinte discussão descreve a diferença entre o antigo e o novo mecanismo de senha, e o que você deve fazer se você atualizar o seu servidor para a versão 4.1 mas precizar de manter compatibilidade com clientes pre-4.1.

Nota: Esta discussão contrasta no comportamento da versão 4.1 com o comportamento da pre-4.1, mas o da versão 4.1 descrito aqui começa relamente na versão 4.1.1. O MySQL é uma distribuição "disferente" porque ela tem um mecanismo um pouco diferente daquele implementado na 4.1.1 e acima. Diferenças entre a versão 4.1.0 e as versões mais recentes são descritas posteriormente.

Antes do MySQL 4.1, o hash de senha calculado pela função `PASSWORD()` tem tamanho de 16 bytes. Este hash se parece com:

```
mysql> SELECT PASSWORD('mypass');
+--------------------+
| PASSWORD('mypass') |
+--------------------+
| 6f8c114b58f2ce9e   |
+--------------------+
```

A coluna `Password` da tabela `user` (na qual estes hashes são armazenados) também têm 16 bytes de tamanho antes do MySQL 4.1.

A partir do MySQL 4.1, a função `PASSWORD()` foi modificada para produzir um valor hash de 41 bytes.

```
mysql> SELECT PASSWORD('mypass');
+-------------------------------------------+
| PASSWORD('mypass')                        |
+-------------------------------------------+
| *43c8aa34cdc98eddd3de1fe9a9c2c2a9f92bb2098d75 |
+-------------------------------------------+
```

De acordo com o mostrado, a coluna `Password` na tabela `user` também deve ter 41 bytes para armazeanar estes valores.

- Se você realiza uma nova instalação do MySQL 4.1, a coluna `Password` será convertida para o tamanho de 41 bytes automaticamente.
- Se você atualizar uma instalação mais antiga para a versão 4.1, você executar o script `mysql_fix_privilege_tables` para atualizar o tamanho da coluna `Password` de 16 para 41 bytes. (O script não altera valores de senhas existentes, que continuam com 16 bytes.)

Uma coluna `Password` mais larga pode armazenar hashes de senha no formato novo e no antigo. O formato de qualquer valor de hash de senha dado podeser determinado de dois modos:

- A diferença óbvia é o tamanho (16 bytes versus 41 bytes)
- A segunda diferença é que os hashes de senha no novo formato sempre começam com um caracter '`*`', que as senhas no formato antigo nunca faziam.

O formato maior do hash de senha tetm melhores propriedades criptográficas, e a autenticação do cliente baseada em hashs mais longos é mais segura que aquela baseada nos antigos hashes menores.

A diferença entre os hashs de senhas menores e maiores são relevantes em como o servidor usa as senhas durante a autenticação e como ela gera hash de senhas para clientes conectados que realizam operações de alteração de senha.

O modo no qual o servidor usa o hash de senha durante a autenticação é afetada pela largura da coluna Password:

- Se a coluna não for larga, apenas a autenticação de hash curto é usada.
- Se a coluna é larga, ela pode guardar tanto hash curtas quanto hashs longas, e o servidor pode usar ambos os formatos:
  - Clientes pre-4.1 podem conectar, mas como els só conhecem o mecanismo hash antigo, eles só podem se conectar pelas contas com hashes curtos.
  - Clientes 4.1 podem autenticar contas com hashes longos ou curtos.

Para contas com o hash curto, o processo de autenticação é na verdade um pouco mais seguro para clientes 4.1 que para clientes mais antigos. Em termos de segurança, o gradiente do menos para o mais seguro é:

- Clientes pre-4.1 autenticando em contas com hash de senha curto
- Clientes 4.1 autenticando em contas com hash de senha curto
- Clientes 4.1 autenticando em contas com hash de senha longo

O modo no qual o servidor gera hashes de senhas para clientes conectados é afetado pela largura da coluna `Password` e pela opção `--old-passwords`. Um servidor 4.1 gera hashes longos apenas se certas condicões forem encontradas: A coluna `Password` deve ser grande o suficiente para armazenar valores longos e a opção `--old-passwords` não deve ser dada. Estas condições se aplicam da seguinte forma:

- A coluna `Password` deve ser grande o suficiente para armazenar hashes longos (41 bytes). Se a coluna não foi atualizada e ainda tem a largura de 16 bytes (antes da 4.1), o servidor avisa que o hash não pode caber nela e gera apenas hashes curtos quando um cliente realiza a operação de troca de senha usando `PASSWORD()`, `GRANT`, ou `SET PASSWORD`. (Este comportamento ocoree se você tiver atualizado para a versão 4.1 mas não executou o script `mysql_fix_privilege_tables` para aumentar a coluna `Password`.)
- Se a coluna `Password` for larga, ela poderá aramazenar tanto os hashes de senha curtos quanto os longos. Neste caso, `PASSWORD()`, `GRANT`, e `SET PASSWORD` irão gerar hashes longos a menos que o servidor tenha sido iniciado com a opção `--old-passwords`. Esta opção força o servidor a gerar hashes de senha curtos.

O propósito da opção `--old-passwords` é permitir que você mantenha compatibilidade com clientes com versões anteriores à 4.1 sob circunstâncias nas quais os servidores gerariam hashes de senha longos. Ele não afeta a autenticação (clientes 4.1 podem ainda usar contas que possuem hash de senha longo), mas ele não previne a criaçõa de um hash de senha longo na tabela `user` como resultado de uma operação de troca de senha. Onde isto ocorrer, a conta não mais poderá ser usada por clientes pré-4.1. Se a opção `--old-passwords`, o seguinte cenário é possível:

- Um cliente antigo conecta a uma conta que têm um hash de senha curto.
- O cliente altera a senha das contas. Sem `--old-passwords`, isto resulta na conta que têm um hash de senha longo.

- A próxima vez que o cliente antigo tentar se conectar à conta, ele não conseguirá, porque a conta agora exige o novo mecanismo de hash durante a autenticação. (Uma vez que uma conta tem um hash de senha longo na tabela de usuário, apenas os clientes 4.1 poderão ser autenticados, porque clientes de versões anteriores a 4.1 não entendem o hash longo.)

Este cenário mostra que é perigoso executar um servidor 4.1 sem usar a opção `--old-passwords`, operações de alteração de senha não irão gerar hashes de senha longos e assim não faz com que as contas se tornem inacessíveis para clientes mais antigos. (Estes clientes não podem bloquear eles mesmos inadivertidamente alterando suas senhas e ficando com um hash de senha longo.

A desvantagem da opção `--old-passwords` é que qualquer senha que você criar ou alterar usará hashes curtos, mesmo para clientes 4.1. Assim, você perde a segurança adicional fornecida pelos hashes de senha longos. Se você quiser criar uma conta qye tenha um hash longo (por exemplom parr uso pelos clientes 4.1), você deve fazê-lo enquanto executa o servidor sem a opção `--old-passwords`.

Os seguintes cenários são possíveis para executar um servidor 4.1:

Cenario 1) Coluna `Password` menor na tabela de usuários

- Apenas hashes curtos podem ser armazenados na coluna `Password`.
- O servidor usa apenas hasghes curtos durante a autenticação do cliente.
- Para clientes conectados, operações de geração de hash de senha envolvendo `PASSWORD()`, `GRANT` ou `SET PASSWORD` usa hashes curtos exclusivamebnte. Qualquer alteração a senha de uma conta faz com que a conta tenha um hash de senha curto.
- A opção `--old-passwords` pode ser usada mas é superflua porque com uma coluna `Password` menor, o servidor irá gerar hashes de senha curtos de qualquer forma.

Cenário 2) Colunas `Password` longas; servidor não iniciado com a opção `--old-passwords`

- Hashes de senha longos e curtos podem ser armazenados na coluna `Password`.
- Clientes 4.1 podem autenticar contas com hashes curtos ou longos.
- Clientes anteioriores ao 4.1 só podem autenticar contas com hash curto.
- Para clientes conectados, operações de geração de hash de senha envolvendo `PASSWORD()`, `GRANT`, ou `SET PASSWORD` usam hashes longos exclusivamente. Qualquer mudança na senha de uma conta fará com que ela possua um hash de senha longo.
- `OLD_PASSWORD()` pode ser usado para gerar explicitamente um hash curto. Por exemplo, para atribuir uma senha curta a uma conta, use `UPDATE` da seguinte forma:

```
mysql> UPDATE user SET Password = OLD_PASSWORD('mypass')
    -> WHERE Host = 'some_host' AND User = 'some_user';
mysql> FLUSH PRIVILEGES;
```

Como indicado anteriormente, o perigoso neste cenário é que é possível que contas com hashes de senha curtos se tornem inacessíveis para cliente anteriores ao 4.1. Qualquer alteração a senha de uma conta feita via `GRANT`, `SET PASSWORD`, ou `PASSWORD()` faz com que a conta tenha um hash de senha longo, e a partir deste ponto, nenhum cliente anterior ao 4.1 poderá autenticar esta conta até que ele seja atualizado para a versão 4.1.

Cenário 3) Coluna `Password` longa; servidor iniciado com a opção `--old-passwords`

- Hashes longos e curtos podem ser armazenados na coluna `Password`.
- Clientes 4.1 podem autenticar contas que tenham hashes longos ou curtos (mas note que é possível criar hashes longos apenas quando o servidor é iniciado sem `--old-passwords`).
- Clientes anteriores ao 4.1 podem autentticar apenas contas com hashes curtos.
- Para clientes conectados, operações de geração de hash de senha envolvendo `PASSWORD()`, `GRANT`, ou `SET PASSWORD` usa hashes curtos exclusivamente. Qualquer alteração em uma senha de conta faz com que a conta tenha um hash de senha curto.

Neste cenário, você não pode criar contas que tenham hashes de senha longo, porque `--old-passwords` previne a criação de hashes longos. Também, se você criar uma conta com um hash longo antes de usar a opção `--old-passwords`, alterar a senha da conta enquanto `--old-passwords` está funcionando faz com que seja dada a conta uma sena curta, fazendo com que ela perca os benefícios de segurança de um hash longo.

As disvantagens para este cenário pode ser resumido como a seguir:

Cenário 1) Você não pode tirar vantagem do hash longo que fornece mais autenticação segura.

Cenário 2) Contas com hashes curtos tornam clientes anteriores ao 4.1 inacessíveis se você alterar a senha deles sem usar `OLD_PASSWORD()` explicitamente.

Cenário 3) `--old-passwords` evita que as contas com hashes curtos se tornem inacessíveis, mas operações de alteração de senhas fazem com que as contas com hashes longos seja revertida para hashes curtos, e você não pode alterá-las de volta para hashes longos enquanto `--old-passwords` está em efeito.

Implicações de Alteração de Hashes de Senha para Aplicativos

Um atualização para o MySQL 4.1 para trazer problemas de compatibilidade para aplicações que usam `PASSWORD()` para gerar senha para os seus próprios propósitos. (Aplicativos não devem fazer isto, porque `PASSWORD()` deve ser usado paenas para gerenciar contas do MySQL. Mas algumas aplicações usam `PASSWORD()` para seus próprios propósitos.) Se você atualizar para o MySQL 4.1 e executar o servidor sob condições onde ele gera hashes de senha longo, uma aplicação que usa `PASSWORD()` para as suas próprias senhas irá falhar. O curso de ação recomendado é modificar o aplicativo para usar outras funções como `SHA1()` ou `MD5()` para produzir valores de hash. Se isto não for possível você pode utilizar a função `OLD_PASSWORD()`, que é fornecida para gerar hashes curtos no formato antigo. (Mas note que `OLD_PASSWORD()` pode vir a não ser mais suportado.)

Se o servidor está rodando sob circuntâncias onde ele gera hashes de senha curtos, `OLD_PASSWORD()` está disponível mas é equivalente a `PASSWORD()`.

Hash de senhas no MySQL 4.1.0 difere do hash no 4.1.1 e acima. As diferenças da versão 4.1.0 são as seguintes:

- Hashes de senhas de 45 bytes em vez de 41 bytes.
- A função `PASSWORD()` não é repetitível. Isto é, com um dado argumento `X`, successivas chamadas a `PASSWORD(X)` geram diferentes resultados.

4.3.12 Causas dos Erros de `Accesso Negado`

Se você encontrar erros de `Accesso Negado` (Access denied) quando tentar conectar-se ao servidor MySQL, a lista abaixo indica alguns caminhos que você pode seguir para corrigir o problema:

- Depois de instalar o MySQL, você executou o script `mysql_install_db` para configurar o conteúdo inicial das tabelas de permissões ? Se não, faça isto. Veja Seção 4.4.4 [Default privileges], Página 260. Testes os privilégios iniciais executando este comando:

  ```
  shell> mysql -u root test
  ```

  O servidor deve deixar você conectar sem erros. Você também deve assegurar que exista o arquivo '`user.MYD`' no diretório do banco de dados do MySQL. Normalmente ele fica em '`CAMINHO/var/mysql/user.MYD`'. onde `CAMINHO` é o caminho para a raiz da instalação do MySQL.

- Depois de terminar uma instalação, você deve conectar ao servidor e configurar seus usuários e suas permissões de acesso.

  ```
  shell> mysql -u root mysql
  ```

  O servidor deve permitir a conexão pois o usuário `root` MySQL vem inicialmente configurado sem senha. Isto também é um risco de segurança, portanto configurar a senha do usuário `root` é algo que deve ser feito enquanto você configura os outros usuários do MySQL.

  Se você tentar se conectar como `root` e obter este erro:

  ```
  Access denied for user: '@unknown' to database mysql
  ```

  isto significa que você não possui um registro na tabela `user` com o valor `'root'` no campo `User` e que o `mysqld` não pode rsolver o nome de máquina do cliente. Neste caso, você deve reiniciar o servidor com a opção `--skip-grant-tables` e editar seu arquivo '`/etc/hosts`' ou o '`\Windows\hosts`' para adicionar uma entrada para sua máquina.

- Se você obter um erro como o seguinte:

  ```
  shell> mysqladmin -u root -pxxxx ver
  Access denied for user: 'root@localhost' (Using password: YES)
  ```

  Significa que você está usando uma senha incorreta. Veja Seção 4.4.8 [Passwords], Página 266.

  Se você esqueceu a senha de root, você pode reiniciar o `mysqld` com a opção `--skip-grant-tables` para alterar a senha. Veja Seção A.4.2 [Resetting permissions], Página 927.

  Se você obter o erro acima mesmo se não tiver configurado uma senha, significa que você tem algum arquivo `my.ini` configurado para passar alguma senha incorreta. Veja Seção 4.1.2 [Option files], Página 216. Você pode evitar o uso de arquivos de opções com a opção `--no-defaults`, como a seguir:

  ```
  shell> mysqladmin --no-defaults -u root ver
  ```

- Se você atualizou uma instalação existente do MySQL de um versão anterior à versão 3.22.11 para a Versão 3.22.11 ou posterior, você executou o script `mysql_fix_privilege_tabels` ? Se não faça isto. A estrutura das tabelas de permissões alteraram com a Versão 3.22.11 do MySQL quando a instrução `GRANT` se tornou funcional. Veja Seção 2.5.6 [Upgrading-grant-tables], Página 130.

- Se os seus privilégios parecerem alterados no meio de uma sessão, pode ser que o superusuário os alterou. A recarga das tabelas de permissões afeta novas conexões dos clientes, mas ela também afeta conexões existentes como indicado em Seção 4.4.3 [Privilege changes], Página 260.
- Se você não consegue fazer a sua senha funcionar, lembre-se que você deve usar a função `PASSWORD()` se você configurar a senha com instruções `INSERT`, `UPDATE` ou `SET PASSWORD`. A função `PASSWORD()` é desnecessária se você especificar a senha usando a instrução `GRANT ... IDENTIFIED BY` ou o comando `mysqladmin password`. Veja Seção 4.4.8 [Senhas], Página 266.
- `localhost` é um sinônimo para seu nome de máquina local, e é também a máquina padrão em que clientes tentam se conectar se você não especificar explicitamente o nome da máquina. Entretanto, conexões para `localhost` não funcionam se você estiver executando em um sistema que utilize MIT-pthreads (conexões `localhost` são feitas utilizando sockets Unix, que não são suportadas pelas MIT-pthreads). Para evitar este problema nestes sistemas, você deve utilizar a opção `--host` para nomear explicitamente o servidor. Isto fará uma conexão TCP/IP ao servidor `myssqld`. Neste caso, você deve ter seu nome de máquina real nos registros da tabela `user` no servidor. (Isto é verdadeiro mesmo se você estiver executando um programa cliente na mesma máquina que o servidor.)
- Se você obter o erro `Access denied` quando tentando conectar ao banco de dados com `mysql -u nome_usuário _nome_bd`, você pode ter um problema com a tabela `user`. Verifique isto executando `mysql -u root mysql` e usando esta sentença SQL:

  ```
  mysql> SELECT * FROM user;
  ```

  O resultado deve incluir uma entrada com as colunas `Host` e `User` combinando com o nome de seu computador e seu nome de usuário no MySQL.
- A mensagem de erro `Access denied` irá dizer a você com qual usuário você está tentando se logar, a máquina que está tentando conectar e se você está utilizando uma senha ou não. Normalmente, você deve ter um registro na tabela `user` que combine exatamente com o nome de máquina e o nome de usuário que forem fornecidos na mensagem de erro. Por exemplo, se você obter uma mensagem de erro que contenha `Using password: NO`, isto significa que você está tentando se conectar sem uma senha.
- Se você obter o seguinte erro quando estiver tentando conectar de uma máquina diferente da que o servidor MySQL estiver executando, então não deve existir um registro na tabela `user` que combine com esta máquina:

  ```
  Host ... is not allowed to connect to this MySQL server
  ```

  Você pode corrigir isto utilizando a ferramenta de linha de comando `mysql` (no servidor!) para adicionar um registro à tabela `user`, `db` ou `host` para coincidir com o usuário e nome de máquina de onde você está tentando conectar, depois execute o comando `mysqladmin flush-privileges`. Se você não estiver executando o MySQL Versão 3.22 e você não sabe o número IP ou o nome da máquina da qual estiver conectando, você deve colocar uma entrada com o valor `'%'` na coluna `Host` da tabela `user` e reiniciar o `mysqld` com a opção `--log` na máquina onde é executado o servidor. Depois tente conectar a partir da máquina cliente, a informação no log do MySQL irá indicar como você está realmente conectando. (Então troque o `'%'` na tabela `user` com o nome da máquina mostrado pelo log. De outra forma você teria um sistema que seria inseguro.)

Outra razão para este erro no Linux pode ser porque você está utilizando uma versão binária do MySQL que é compilada com uma versão diferente da glibc que você está usando. Neste caso você deve atualizar seu SO/Glibc ou fazer o download da versão fonte do MySQL e compilá-la. Um RPM fonte é, normalmente, fácil de compilar e instalar, logo, isto não é um grande problema.

- Se você obter uma mensagem de erro onde o nome da máquina não é exibido ou, no lugar do nome da máquina existir um IP, mesmo se você tenta a conexão com um nome de máquina:

```
shell> mysqladmin -u root -pxxxx -h some-hostname ver
Access denied for user: 'root@' (Using password: YES)
```

  Isto significa que o MySQL obteve algum erro quando tentava resolver o IP para um nome de maquina. Neste caso você pode executar `mysqladmin flush-hosts` para zerar o cache DNS interno. Veja Seção 5.5.5 [DNS], Página 459.

  Algumas soluções permanentes são:

  - Tente descobrir o que está errado com seu servidor DNS e corrija os erros.
  - Especifique números IPs no lugar de nomes nas tabelas de privilégios do MySQL.
  - Inicie o `mysqld` com `--skip-name-resolve`.
  - Inicie o `mysqld` com `--skip-host-cache`.
  - Conecte à `localhost` se você estiver executando o servidor e o cliente na mesma máquina.
  - Coloque os nomes das máquinas clientes em `/etc/hosts`.

- Se `mysql -u root test` funciona mas `mysql -h nome_servidor -u root test` resultar em `Access denied`, então você pode não ter o nome correto para a sua máquina na tabela `user`. Um problema comum é quando o valor de `Host` na entrada da tabela user especifica um nome de máquina não qualificado, mas as rotinas de resolução de nomes de seu sistema retornam um nome qualificado completo do domínio (ou vice-versa). Por exemplo, se você tem uma entrada com o nome `'tcx'` na tabela `user`, mas seu DNS diz ao MySQL que o nome da máquina é `'tcx.subnet.se'`, a entrada não irá funcionar. Tente adicionar um registro à tabela `user` que contenha o número IP de sua máquina como o valor da coluna `Host`. (Uma alternativa, seria adicionar um registro à tabela `user` com o valor de `Host` contendo um metacaracter, por exemplo, `'tcx.%'`. Entretanto, o uso de nomes de máquinas terminando com '`%`' é *inseguro* e *não* é recomendado!)
- Se `mysql -u nome_usuário test` funciona mas `mysql -u nome_usuário outro_bd` não funconar, você não possui uma entrada para `outro_bd` listado na tabela `db`.
- Se `mysql -u nome_usuário nome_bd` funciona quando executado no próprio servidor, mas `mysql -u nome_máquina -u nome_usuário nome_bd` não funciona quando executado em outra máquina cliente, você não possui o nome da máquina cliente listado na tabela `user` ou na tabela `db`.
- Se você não estiver entendendo porque obtem `Access denied`, remova da tabela `user` todas as entradas da coluna `Host` que contenham meta caracteres (entradas que contenham '`$`' ou '`_`'). Um erro muito comum é inserir uma nova entrada com `Host='%'` e `User='algum usuário'`, pensando que isto irá permitir a você especificar `localhost` para conectar da mesma máquina. A razão disto não funcionar é que os privilégios

padrões incluem uma entrada com `Host='localhost'` e `User=''`. Como esta entrada tem o valor `'localhost'` em `Host` que é mais específica que `'%'`, ela é usada no lugar da nova entrada quando se conectar de `localhost`! O procedimento correto é inserir uma segunda entrada com `Host='localhost'` e `User='algum_usuário'`, ou remover a entrada com `Host='localhost'` e `User= ''`.

- Se você obter o seguinte erro, você pode ter um problema com a tabela `db` ou a tabela `host`:

  ```
  Access to database denied
  ```

  Se a entrada selecionada da tabela `db` tiver um valor vazio na coluna `Host`, tenha certeza que exista uma ou mais entradas correspondentes na tabela `host` especificando quais máquinas aplicam-se à tabela `db`.

  Se você obter o erro quando estiver utilizando comandos SQL `SELECT ... INTO OUTFILE` ou `LOAD DATA INFILE`, a entrada na tabela `user` provavelmente não tem o privilégio **file** habilitado.

- Lembre-se que programas clientes irão usar parâmetros de conexões especificados em arquivos de configuração ou variáveis ambientais. Veja Apêndice F [Environment variables], Página 1092. Se parecer que algum cliente está enviando parâmetros errados para a conexão e você não os especificou na linha de comando, verifique seu ambiente e o arquivo '`.my.cnf`' no seu diretório home. Você pode também conferir os arquivos de configurações do servidor MySQL, apesar de não ser interessante gravar configurações de cliente nestes arquivos. Veja Seção 4.1.2 [Option files], Página 216. Se você obter a mensagem de acesso negado (`Access denied`) quando estiver executando um cliente sem opções, tenha certeza que você não especificou uma senha antiga em nenhum de seus arquivos de opções! Veja Seção 4.1.2 [Option files], Página 216.

- Se você fizer alterações para as tabelas de permissões diretamente (utilizando uma instrução `INSERT` ou `UPDATE`) e suas alterações parecem ser ignoradas, lembre que você deve usar uma instrução `FLUSH PRIVILEGES` ou executar um comando `mysqladmin flush-privileges` para o servidor ler novamente as tabelas com os privilégios. De outra forma, suas alterações não farão efeito até que o servidor seja reiniciado. Lembre-se que depois de configurar a senha de `root` com um comando `UPDATE`, não será necessário especificar a senha até que você atualize os privilégios, pois o servidor ainda não saberá que você alterou a senha!

- Se você tiver problemas de acesso com Perl, PHP, Python ou um programa ODBC, tente conectar ao servidor com `mysql -u nome_usuário nome_bd` ou `mysql -u nome_usuário -psua_senha nome_bd`. Se você consegue conectar com o cliente `mysql`, existe algum problema com seu programa e não o acesso aos privilégios (Note que não espaço entre `-p` e a senha; você também pode utilizar a sintaxe `--password=sua_senha` para especificar a senha. Se você utilizar a opção `-p` sozinha, o MySQL irá lhe solicitar a senha.)

- Para testar, iniciae o daemon `mysqld` com a opção `--skip-grant-tables`. Então você pode alterar as tabelas de permissões do MySQL e utilizar o script `mysqlaccess` para conferir se suas modificações fizeram o não o efeito desejado. Quando você estiver satisfeito com suas alterações, execute `mysqladmin flush-privileges` para dizer ao servidor `mysqld` para iniciar utilizando as novas tabelas com os privilégios. **Nota:** Recarregar as tabelas de permissões sobrescreve a opção `--skip-grant-tables`. Isto

lhe permite dizer ao servidor para começar a utilizar as tabelas de permissões novamente sem reiniciá-lo.

- Se tudo mais falhar, inicie o servidor `mysqld` com uma opção de depuração (por exemplo, `--debug=d,general,query`). Isto irá imprimir informações de máquinas e usuários sobre tentativas de conexões, e também informações sobre cada comando disparado. Veja Seção E.1.2 [Making trace files], Página 1080.
- Se você tiver outros problemas com as tabelas de permissões do MySQL e sente que deve enviar o problema para a lista de discussão, sempre forneça um descarga das tabelas de permissões do seu MySQL. Você pode descarregar as tabelas com o comando `mysqldump mysql`. Como sempre, envie seus problemas utilizando o script `mysqlbug`. Veja Seção 1.7.1.3 [Bug reports], Página 36. Em alguns casos você pode precisar reiniciar o `mysqld` com a opção `--skip-grant-tables` para executar o `mysqldump`.

4.4 Gerenciamento das Contas dos Usuários no MySQL

4.4.1 A Sintaxe de `GRANT` e `REVOKE`

```
GRANT priv_type [(column_list)] [, tipo_priv [(column_list)] ...]
    ON {tbl_name | * | *.* | db_name.*}
    TO user_name [IDENTIFIED BY [PASSWORD] 'password']
        [, user_name [IDENTIFIED BY [PASSWORD] 'password'] ...]
    [REQUIRE
        NONE |
     [{SSL| X509}]
[CIPHER cipher [AND]]
[ISSUER issuer [AND]]
[SUBJECT subject]]
    [WITH [GRANT OPTION | MAX_QUERIES_PER_HOUR # |
                          MAX_UPDATES_PER_HOUR # |
                          MAX_CONNECTIONS_PER_HOUR #]]

REVOKE priv_type [(column_list)] [, priv_type [(column_list)] ...]
    ON {tbl_name | * | *.* | db_name.*}
    FROM user_name [, user_name ...]
```

O comando `GRANT` é implementado no MySQL versão 3.22.11 ou posterior. Para versões anteriores do MySQL, a instrução `GRANT` não faz nada.

Os comandos `GRANT` e `REVOKE` permitem aos administradores do sistema criar usuários e conceder e revogar direitos aos usuários do MySQL em quatro níveis de privilégios:

Nível Global

Privilégios globais aplicam para todos os bancos de dados em um determinado servidor. Estes privilégios são armazenados na tabela `mysql.user`. `GRANT ALL ON *.*` e `REVOKE ALL ON *.*` concederão e revogarão apenas privilégios globais.

Nível dos bancos de dados
: Privilégios de bancos de dados aplicam-se a todas as tabelas em um determinado banco de dados. Estes privilégios são armazenados nas tabelas `mysql.db` e `mysql.host`. `GRANT ALL ON db.*` e `REVOKE ALL ON db.*` concederão e revogarão apenas privilégios de banco de dados.

Nível das tabelas
: Privilégios de tabelas aplicam-se a todas as colunas em uma determinada tabela. Estes privilégios são armazenados na tabela `mysql.tables_priv`. `GRANT ALL ON db.table` e `REVOKE ALL ON db.table` concederão e revogarão apenas privilégios de tabelas.

Nível das colunas
: Privilégios de colunas aplicam-se a uma única coluna em uma determinada tabela. Estes privilégios são armazenados na tabela `mysql.columns_priv`.

Para as instruções `GRANT` e `REVOKE`, `tipo_priv` pode ser especificado como um dos seguintes:

`ALL [PRIVILEGES]`	Configura todos os privilégios simples exceto `WITH GRANT OPTION`
`ALTER`	Permite o uso de `ALTER TABLE`
`CREATE`	Permite o uso de `CREATE TABLE`
`CREATE TEMPORARY TABLES`	Permite o uso de `CREATE TEMPORARY TABLE`
`DELETE`	Permite o uso de `DELETE`
`DROP`	Permite o uso de `DROP TABLE`.
`EXECUTE`	Permite que o usuário execute stored procedures (MySQL 5.0)
`FILE`	Permite o uso de `SELECT ... INTO OUTFILE` e `LOAD DATA INFILE`.
`INDEX`	Permite o uso de `CREATE INDEX` e `DROP INDEX`
`INSERT`	Permite o uso de `INSERT`
`LOCK TABLES`	Permite o uso de `LOCK TABLES` em tabelas nas quais se tem o privilégio `SELECT`.
`PROCESS`	Permite o uso de `SHOW FULL PROCESSLIST`
`REFERENCES`	Para o futuro
`RELOAD`	Permite o uso de `FLUSH`
`REPLICATION CLIENT`	Da o direto ao usuário de perguntar onde o slave/master está.
`REPLICATION SLAVE`	Necessário para a replicação dos slaves (para ler logs binário do master).
`SELECT`	Permite o uso de `SELECT`
`SHOW DATABASES`	`SHOW DATABASES` exibe todos os banco de dados.
`SHUTDOWN`	Permite o uso de `mysqladmin shutdown`
`SUPER`	Permite a conexão (uma vez) mesmo se max_connections tiverem sido alcançados e executa o comando `CHANGE MASTER`, `KILL thread`, `mysqladmin debug`, `PURGE MASTER LOGS` e `SET GLOBAL`
`UPDATE`	Permite o uso de `UPDATE`
`USAGE`	Sinônimo para "sem privilégios."
`GRANT OPTION`	Sinônimo para `WITH GRANT OPTION`

`USAGE` pode ser usado quando você quer criar um usuário sem privilégios.

Os privilégios `CREATE TEMPORARY TABLES`, `EXECUTE`, `LOCK TABLES`, `REPLICATION ...`, `SHOW DATABASES` e `SUPER` são novos na versão 4.0.2. Para usar estes novos privilégios após atualizar para 4.0.2, você tem que executar o script `mysql_fix_privilege_tables`. Veja Seção 2.5.6 [Upgrading-grant-tables], Página 130.

Em versões anteriores do MySQL, o privilégio `PROCESS` dá o mesmo direitos que o novo privilégio `SUPER`.

Para anular o privilégio **grant** de um usuário, utilize o valor `tipo_priv` de `GRANT OPTION`:

```
mysql> REVOKE GRANT OPTION ON ... FROM ...;
```

Os únicos valores de `tipo_priv` que você pode especificar para uma tabela são `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `DROP`, `GRANT`, `INDEX` e `ALTER`.

Os únicos valores de `tipo_priv` que você pode especificar para uma coluna (isto é, quando você usar uma cláusula `column_list`) são `SELECT`, `INSERT` e `UPDATE`.

O MySQL permite que você crie privilégios a nível de banco de dados mesmo se o banco de dados não existir para tornar fácil de se preparar para o uso do banco de dados. Atualmente, no entanto, o MySQL não permite criar permissões de a nível de tabela se a tabela não existir. O MySQL não revogará automaticamente qualquer privilégio, mesmo se você apagar uma tabela ou banco de dados.

Você pode configurar privilégios globais utilizando a sintaxe `ON *.*`. Você pode configurar privilégios de bancos de dados utilizando a sintaxe `ON nome_bd.*`. Se você especificar `ON *` e estiver com algum banco de dados aberto, será configurado os privilégios somente para este banco de dados. (**AVISO:** Se você especificar `ON *` e você *não* tem possui um banco de dados aberto, irá afetar os privilégios globais!).

Note por favor Os metacaracteres '`_`' e '`%`' são permitidos na especificação dos nomes de bancos de dados em comandos `GRANT`. Isto significa que se você deseja usar um caracater '`_`' como parte de um nome de banco de dados, você deve especificá-lo como '`\_`' no comando `GRANT`, para prevenir o usuário de poder acessar bancos de dados adicionais que correspondam ao padrão do metacaracter, ex., `GRANT ... ON 'foo\_bar'.* TO ...`.

Para acomodar concessões de direitos para usuários de máquinas arbitrárias, o MySQL suporta a especificação do valor `user_name` no formato `usuário@máquina`. Se você desejar especificar uma string `user` contendo caracteres especiais (como o '`-`'), ou uma string contendo caracteres especiais ou meta caracteres (como o '`%`'), você pode colocar o usuário ou o nome de máquina entre aspas (por exemplo, `'usuário-teste'@'máquina-teste'`).

Você pode especificar meta caracteres no nome da máquina. Por exemplo, `user@"%.loc.gov"` se aplica a `user` para qualquer máquina no domínio `loc.gov`, e `user@"144.155.166.%"` se aplica a `user` em qualquer máquina na subrede de classe C `144.155.166`.

O formato simples `user` é sinônimo de `user@"%"`.

O MySQL não suporta metacaracteres em nomes de usuários. Usuários anônimos são definidos inserindo entradas com `User=''` na tabela `mysql.user` ou criando um usuário com um nome vazio com o comando `GRANT`.

Nota: Se você permite o acesso de usuários anônimos ao seu servidor MySQL, você deve também concecder privilégios a todos os usuários locais como `user@localhost` porque, de outra forma, a entrada de usuário anônimo para a máquina local na tabela `mysql.user` será usada quando o usuário tentar a conexão ao servidor MySQL da máquina local!

Você pode verificar se isto se aplica a você executando a seguinte instrução:

```
mysql> SELECT Host,User FROM mysql.user WHERE User='';
```

No momento, `GRANT` suporta somente nomes de máquinas, tabelas bancos de dados e colunas até 60 caracteres. Um nome de usuário pode ter até 16 caracteres.

Os privilégios para uma tabela ou coluna são formados através do OU lógico dos privilégios em cada um dos quatro níveis de privilégios. Por exemplo, se a tabela `mysql.user` especifica que um usuário tem um privilégio global **select**, isto não pode ser negado por uma entrada no nível de banco de dados, tabela ou coluna.

Os privilégios para uma coluna podem ser calculados da seguinte forma:

```
privilégios globais
OR (privilégios de banco de dados AND privilégios de máquina)
OR privilégios de tabela
OR privilégios de coluna
```

Na maioria dos casos, os direitos a um usuário são atribuídos em apenas um dos níveis de privilégios, portanto a vida normalmente não é tão complicada como mostrado acima. Os detalhes do procedimento de verificação dos privilégios são apresentados em Seção 4.3 [Sistema de privilégios], Página 226.

Se você concede privilégios para uma combinação de usuário e máquina que não existem na tabela `mysql.user`, um registro é adicionado e permanece lá até ser removido com um comando `DELETE`. Em outras palavras, `GRANT` pode criar registros na tabela `user`, mas `REVOKE` não as removerá; para removê-las você deve usar a instrução explícita `DELETE`.

Na Versão 3.22.12 ou posterior do MySQL, se um novo usuário é criado ou se você possui privilégios de concessão globais, a senha do usuário será especificada utilizando a cláusula `IDENTIFIED BY`, se uma for dada. Se o usuário já possui uma senha, ela é trocada pela nova.

Se você não quiser enviar a senha em texto puro você pode usar a opção `PASSWORD` seguido de uma senha embaralhada da função SQL `PASSWORD()` ou da função da API C `make_scrambled_password(char *to, const char *password)`.

CUIDADO: Se você criar um novo usuário mas não especificar uma cláusula `IDENTIFIED BY`, o usuário não possuirá uma senha. Isto não é seguro.

Senhas podem também ser configuradas com o comando `SET PASSWORD`. Veja Seção 5.5.6 [SET OPTION], Página 460.

Se você conceder privilégios para um banco de dados, uma entrada na tabela `mysql.db` é criada se necessário. Quando todos os privilégios para o banco de dados forem removidos com `REVOKE`, este registro é removido.

Se um usuário não tem privilégios em uma tabela, a tabela não é mostrada quando o usuário solicita uma lista de tabelas (com a instrução `SHOW TABLES` por exemplo). O mesmo é verdade para `SHOW DATABASES`

A cláusula `WITH GRANT OPTION` dá ao usuário habilidade de fornecer à outros usuários quaisquer privilégios que ele tenha em um nível específico de privilégio. Você deve ter cuidado ao fornecer o privilégio **grant**, pois dois usuários podem se unir para unir privilégios!

`MAX_QUERIES_PER_HOUR #`, `MAX_UPDATES_PER_HOUR #` e `MAX_CONNECTIONS_PER_HOUR #` sãp novos no MySQL versão 4.0.2. Estas opções limitam o número de consultas/atualizações e logins que o usuários pode fazer durente uma hora. Se `#` é 0 (padrão), então isto significa que

não há limites para aquele usuário. Veja Seção 4.4.7 [Recursos do usuários], Página 265. Nota: para especificar qualquer destas opções para um usuário existente sem adicionar outros privilégios adicionais, use `GRANT USAGE ON *.* ... WITH MAX_...`.

Você não pode conceder a outro usuário um privilégio que não possua; o privilégio **GRANT** possibilita fornecer somente os privilégios que possuir.

Esteja ciente que quando conceder a um usuário o privilégio **GRANT** em um nível particular de privilégios, qualquer privilégio que o usuário já possua (ou seja fornecido no futuro!) nesse nível também pode ser concedido por este usuário. Suponha que você conceda a um usuário o privilégio **INSERT** em um banco de dados. Se você conceder o privilégio **SELECT** no banco de dados e especificar `WITH GRANT OPTION`, o usuário além de poder repassar o privilégio **SELECT** poderá também repassar o **insert**. Se você concede o privilégio **UPDATE** para o usuário no banco de dados, o usuário poderá conceder os privilégios **INSERT**, **SELECT** e **UPDATE**.

Você não deve conceder privilégios **ALTER** a um usuário comum. Se você fizer isto, o usuário pode tentar enganar o sistema de privilégios renomeando tabelas!

Perceba que se você estiver utilizando privilégios de tabelas ou colunas, mesmo que para apenas um usuário, o servidor examina os privilégios de tabelas e colunas para todos os usuários e isto irá deixar o MySQL um pouco mais lento.

Quando o `mysqld` inicia, todos os privilégios são lidos na memória. Privilégios de bancos de dados, tabelas e colunas são iniciados um vez, e privilégios ao nível de usuário fazem efeito na próxima vez que o usuário conectar. Modificações nas tabelas de permissões que você realiza utilizando `GRANT` ou `REVOKE` são percebidas pelo servidor imediatamente. Se você modificar as tabelas de permissões manualmente (utilizando `INSERT`, `UPDATE`, etc), você deve executar uma instrução `FLUSH PRIVILEGES` ou executar `mysqladmin flush-privileges` para dizer ao servidor para recarregar as tabelas de permissões. Veja Seção 4.4.3 [Privilege changes], Página 260.

As maiores diferenças entre o padrão SQL e versões MySQL de `GRANT` são:

- No MySQL privilégios são fornecidos para uma combinação de usuário e máquina e não somente para um usuário.
- O SQL-99 não possui privilégios no nível global ou de bancos de dados, e não suporta todos os tipos de privilégios que o MySQL suporta. O MySQL não suporta os privilégios `TRIGGER`, `EXECUTE` ou `UNDER` do SQL-99.
- Os privilégios do SQL-99 são estruturadados em uma maneira hierárquica. Se você remover um usuário, todos os privilégios do usuário são removidos. No MySQL os privilégios concedidos não são removidos automaticamente, mas você deve removê-los se necessário.
- Se no MySQL você possuir o privilégio `INSERT` em somente parte das colunas em uma tabela, você pode executar instruções `INSERT` na tabela; As colunas em que você não tem o privilégio `INSERT` irão receber seus valores padrões. O SQL-99 necessita que você tenha o privilégio `INSERT` em todas as colunas.
- Quando você remove uma tabela no SQL-99, todos os privilégios para a tabela são removidos. Se você remover um privilégio no SQL-99, todos os privilégios que foram concedidos baseado neste privilégio são também removidos. No MySQL, privilégios só podem ser removidos com comandos `REVOKE` explícitos ou manipulando as tabelas de permissões do MySQL.

Para uma descrição do uso de `REQUIRE`, veja Seção 4.4.10 [Secure connections], Página 268.

4.4.2 Nomes de Usuários e Senhas do MySQL

Existem várias diferenças entre a forma que nomes de usuários e senhas são usados pelo MySQL e a forma que são usados pelo Unix ou Windows:

- Nomes de usuários, como usado pelo MySQL para propósitos de autenticação, não tem nenhuma relação com os nomes de usuários do Unix (nomes de login) ou nomes de usuários Windows. A maioria dos clientes MySQL, por padrão, tentam se conectar utilizando o nome de usuário atual do Unix como o nome de usuário no MySQL, mas isto existe somente por conveniência. Programas clientes permite especificar um nome diferente com as opções `-u` ou `--user`. Isto significa que você não pode tornar um banco de dados seguro a menos que todos os usuários do MySQL possuam senhas. Qualquer um pode tentar se conectar ao servidor utilizando qualquer nome, e eles se conectarão com qualquer nome que não possua uma senha.
- Nomes de usuários MySQL podem ter o tamanho de até 16 caracteres; Nomes de usuário Unix normalmente são limitados até 8 caracteres.
- Senhas MySQL não tem nenhuma relação com senhas Unix. Não existe nenhuma associação entre a senha em que você utiliza para logar-se a uma máquina Unix e a senha que é utilizada para acessar um banco de dados na mesma máquina.
- O MySQL criptografa senhas utilizando um algorítimo diferente que o utilizado pelo processo de login do Unix. Veja as descrições das funções `PASSWORD()` e `ENCRYPT()` em Seção 6.3.6.2 [Miscellaneous functions], Página 547. Perceba que mesmo que a senha é armazenada 'embaralhada', o conhecimento da sua senha 'embaralhada' é o suficiente para conseguir se conectar ao servidor MySQL!

  A partir da versão 4.1, o MySQL emprega um mecanismo de senha e login diferentes que é seguro mesmo se for feito um sniff no pacote TCP/IP e/ou o banco de dados mysql for capturado.

Usuários MySQL e seus privilégios são criados normalmente com o comando `GRANT`, Veja Seção 4.4.1 [GRANT], Página 254.

Quando você se conecta a um servidor MySQL com um cliente de linha de comando você pode especificar a senha com `--password=sua-senha`. Veja Seção 4.3.8 [Connecting], Página 238.

```
mysql --user=monty --password=guess nome_do_banco
```

Se você deseja que o cliente lhe solicite a senha, deve ser especificado o parâmetro `--password` sem nenhum argumento

```
mysql --user=monty --password nome_do_banco
```

ou no formato curto:

```
mysql -u monty -p nome_do_banco
```

Perceba que no último exemplo a senha **não** é 'nome_do_banco'.

Se você deseja usar a opção `-p` para fornecer uma senha você deve fazer assim:

```
mysql -u monty -pguess database_name
```

Em alguns sistemas, a chamada da biblioteca que é utilizada pelo MySQL para solicitar por uma senha corta automaticamente a senha para 8 caracteres. Internamente o MySQL não limita o tamanho limite da senha.

4.4.3 Quando as Alterações nos Privilégios tem Efeito

Quando o `mysqld` inicia, todas o conteúdo das tabelas de permissões são lidos em memória e tem efeito neste momento.

As modificações das tabelas de permissões que você realiza utilizando `GRANT`, `REVOKE` ou `SET PASSWORD` são imediatamente reconhecidas pelo servidor.

Se você alterar as tabelas de permissões manualmente (utilizando `INSERT`, `UPDATE`, etc), você deve executar a instrução `FLUSH PRIVILEGES` ou executar `mysqladmin flush-privileges` ou `mysqladmin reload` para dizer ao servidor para recarregar as tabelas de permissões. De outra forma suas alterações *não terão efeito* até que o servidor seja reiniciado. Se você alterar as tabelas de permissões manualmente mas se esquecer de recarregar os privilégios, suas alteração vão parecer não ter feito nenhuma diferença!

Quando o servidor reconhecer que as tabelas de permissões foram alteradas, conexões existentes são afetadas da seguinte forma:

- Alterações nos privilégios de tabelas e colunas fazem efeito com a próxima requisição do cliente.
- Alterações nos privilégios de bancos de dados fazem efeito no próximo comando `USE nome_bd`.
- Alterações nos privilégios globais e alterações de senhas fazem efeito na próxima vez que o cliente conectar.

4.4.4 Configurando os Privilégios Iniciais do MySQL

Depois de instalar o MySQL, você configura os privilégios iniciais dos acessos executando `scripts/mysql_install_db`. Veja Seção 2.3.1 [Quick install], Página 95. O script `mysql_install_db` inicia o servidor `mysqld`, depois inicializa as tabelas de permissões com a seguinte configuração dos privilégios:

- O usuário `root` do MySQL é criado como um superusuário que pode fazer qualquer coisa. Conexões devem ser feitas através da máquina local.

  NOTA: A senha inicial de `root` é vazia, portanto qualquer um que conectar como `root` *sem senha* terá direito a todos os privilégios.
- Um usuário anônimo é criado e pode fazer o que desejar com bancos de dados com nome `'test'` ou iniciando com `'test_'`. Conexões devem ser feitas da máquina local. Isto significa que usuários locais podem se conectar sem senha e serem tratados como usuários anônimos.
- Outros privilégios são negados. Por exemplo, usuários normais não podem executar `mysqladmin` ou `mysqladmin processlist`.

NOTA: Os privilégios padrões são diferentes no Windows. Veja Seção 2.1.1.8 [Executando o Windows], Página 68.

Como sua instação inicialmente é parcialmente aberta, uma das primeiras coisas que você deve fazer é especificar uma senha para o usuário `root` do MySQL. Você pode fazer isto como a seguir (perceba que a senha foi especificada utilizando a função `PASSWORD()`):

```
shell> mysql -u root mysql
mysql> SET PASSWORD FOR root@localhost=PASSWORD('nova_senha');
```

Substitua `'nova_senha'` pela senha que você deseja usar.

Se você souber o que esta fazendo, você também pode manipular diretamente a tabela privilégios:

```
shell> mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('nova_senha')
    ->     WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

Outra forma de configurar a senha é utilizando o comando `mysqladmin`:

```
shell> mysqladmin -u root password nova_senha
```

Somente usuários com acesso de escrita/atualização ao banco de dados `mysql` podem alterar a senha de outros usuários. Todos os usuários comuns (não os anônimos) podem alterar somente a própria senha com um dos comandos acima ou com `SET PASSWORD=PASSWORD('nova_senha')`.

Perceba que se você atualizar a senha na tabela `user` diretamente utilizando `UPDATE`, você deve dizer ao servidor para reler as tabelas de permissões (com `FLUSH PRIVILEGES`), de outra forma a alteração não seria notificada.

Uma vez que a senha de `root` foi configurada, você deve informar a senha quando se conectar ao servidor MySQL como `root`.

Você pode desejar deixar a senha de `root` em branco para que você não precise especificá-la quando realizar configurações adicionais ou testes. Entretanto, tenha certeza de configurá-la antes de utilizar sua instalação para qualquer ambiente de produção.

Veja o script `scripts/mysql_install_db` para ver como são configurados os privilégios padrões. Você pode usar isto como uma base para ver como adicionar outros usuários.

Se você deseja que os privilégios iniciais sejam diferentes do descrito acima, é possível modificar o script `mysql_install_db` antes de executá-lo.

Para recriar as tabelas de permissões completamente, remova todos os arquivos '`.frm`' '`.MYI`' e '`.MYD`' no diretório contendo o banco de dados `mysql`. (Este é o diretório chamado '`mysql`' sob o diretório do banco de dados, que é listado quando você executa `mysqld --help`.) Depois execute o script `mysql_install_db`, possivelmente depois de editá-lo para criar os privilégios desejáveis.

NOTA: Para versões do MySQL mais antigas que a versão 3.22.10, você não deve apagar os arquivos '`.frm`'. Se você fizer isso acidentalmente, você deve voltá-los a partir de sua distribuição MySQL antes de executar `mysql_install_db`.

4.4.5 Adicionando Novos Usuários ao MySQL

Existem duas maneiras de adicionar usuários: utilizando instruções `GRANT` ou manipulando as tabelas de permissões do MySQL diretamente. O método preferido é utilizar instruções

`GRANT`, porque elas são mais concisas e menos propensas a erros. Veja Seção 4.4.1 [GRANT], Página 254.

Existem vários programas de colaboradores (como o `phpMyAdmin`) que podem ser utilizados para criar e administrar usuários. Veja Apêndice B [Contrib], Página 940.

Os exemplos abaixo mostram como usar o cliente `mysql` para configurar novos usuários. Estes exemplos assumem que privilégios são configurados de acordo com os padrões descritos na seção anterior. Isto significa que para fazer alterações, você deve se conectar na mesma máquina em que o `mysqld` está executando, você deve se conectar com o usuário `root`, e o usuário `root` deve ter os privilégios **inster** ao banco de dados `mysql` e o administrativo **reload**. Também, se você alterou a senha do usuário `root`, você deve especificá-la para os comandos `mysql` abaixo.

Primeiro, use o programa `mysql` para se conectar ao servidor como o usuário `root` do MySQL:

```
shell> mysql --user=root mysql
```

Você pode adicionar novos usuários utilizando instruções `GRANT`:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@localhost
           IDENTIFIED BY 'alguma_senha' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@'%'
           IDENTIFIED BY 'alguma_senha' WITH GRANT OPTION;
mysql> GRANT RELOAD,PROCESS ON *.* TO admin@localhost;
mysql> GRANT USAGE ON *.* TO dummy@localhost;
```

Estas instruções `GRANT` configuram três novos usuários:

`monty` Um superusuário completo que pode conectar ao servidor de qualquer lugar, mas deve utilizar uma senha `'alguma_senha'` para fazer isto. Perceba que devemos utilizar instruções `GRANT` para `monty@localhost` e `monty@"%"`. Se nós não adicionarmos a entrada com `localhost`, a entrada para o usuário anônimo para `localhost` que é criada por `mysql_install_db` toma precedência quando nos conectarmos da máquina local, porque ele contem um campo `Host` com um valor mais específico e também vem antes na ordenação da tabela `user`.

`admin` Um usuário que possa conectar de `localhost` sem uma senha e que é concedido os privilégios administrativos **reload** e **process**. Isto permite ao usuário a execução dos comandos `mysqladmin reload`, `mysqladmin refresh` e `mysqladmin flush-*`, bem como o `mysqladmin processlist`. Nenhum privilégio a nível de bancos de dados é concedido. (Depois eles podem ser adicionados utilizando instruções `GRANT` adicionais.)

`dummy` Um usuário que pode conectar sem uma senha, mas somente na máquina local. Não são concedidos nenhum privilégio—o tipo de privilégio `USAGE` permite a criação de um usuário sem privilégios. Ele tem o efeito de criar todos os privilégios globais com `'N'`. Considera-se que você irá conceder privilégios específicos a conta posteriormente.

Também é possível adicionar a mesma informação de acesso do usuário diretamente, utilizando instruções `INSERT` e depois dizendo ao servidor para recarregar as tabelas de permissões:

```
shell> mysql --user=root mysql
mysql> INSERT INTO user VALUES('localhost','monty',PASSWORD('alguma_senha'),
                'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user VALUES('%','monty',PASSWORD('alguma_senha'),
                'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user SET Host='localhost',User='admin',
                 Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host,User,Password)
                        VALUES('localhost','dummy','');
mysql> FLUSH PRIVILEGES;
```

Dependendo da sua versão do MySQL, você pode precisar utilizar um número diferente de valores `'Y'` acima. (Versões anteriores à versão 3.22.11 tem menos campos de privilégios, e posteriores a 4.02 têm mais). Para o usuário `admin`, a maior sintaxe legível de `INSERT` usando `SET` que está disponível a partir da versão 3.22.11 é a utilizada.

Note que para configurar um superusuário, você só precisar criar uma entrada na tabela `user` com os campos de privilégios configurados para `'Y'`. Não é necessário gerar entradas nas tabelas `db` ou `host`.

Na última instrução `INSERT` (para o usuário `dummy`), apenas as colunas `Host`, `User` e `Password` nos registros da tabela `user` tem valores atribuídos. Nenhuma das colunas de privilégios são definidas explicitamente, assim o MySQL atribui a todas o valor padrão de `'N'`. Isto é a mesma coisa que o `GRANT USAGE` faz.

O seguinte exemplo adiciona um usuário `custom` que pode acessar o banco de dados `bankaccout` apenas do `localhost`, o banco de dados `expenses` somente de `whitehouse.gov` e o banco de dados `customer` de todas de `server.domain`. Ele deseja utilizar a senha `obscure` das três máquinas.

Para configurar os privilégios deste usuário utilizando instruções `GRANT`, execute estes comandos:

```
shell> mysql --user=root mysql
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON bankaccount.*
    ->     TO custom@localhost
    ->     IDENTIFIED BY 'obscure';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON expenses.*
    ->     TO custom@'whitehouse.gov'
    ->     IDENTIFIED BY 'obscure';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON customer.*
    ->     TO custom@'server.domain'
    ->     IDENTIFIED BY 'obscure';
```

Para configurar os privilégios do usuário modificiando as tabelas de permissões diretamente, utilize estes comandos (perceba o `FLUSH PRIVILEGES` no final):

```
shell> mysql --user=root mysql
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('localhost','custom',PASSWORD('obscure'));
```

```
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('whitehouse.gov','custom',PASSWORD('obscure'));
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('server.domain','custom',PASSWORD('obscure'));
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES
    -> ('localhost','bankaccount','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES
    -> ('whitehouse.gov','expenses','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES('server.domain','customer','custom','Y','Y','Y','Y','Y','Y');
```

Como no exemplo anterior que usaram as instruções `INSERT`, você pode precisar de usar um número diferentes de valores `'Y'`, dependendo de sua versão do MySQL.

As primeiras três instruções `INSERT` adicionam entradas na tabela `user` que permite ao usuário `custom` conectar a partir de várias máquinas com a senha determinada, mas não concede permissões ao mesmo (todos os privilégios são configurados com o valor padrão de `'N'`). As próximas três instruções `INSERT` adicionam entradas na tabela `db` que concedem privilégios à `custom` para os bancos de dados `bankaccount`, `expenses` e `customer`, mas só quando acessados à partir das máquinas apropriadas. Normalmente, depois de modificar as tabelas de permissões diretamente, você deve dizer ao servidor para recarregá-las (com `FLUSH PRIVILEGES`) para que as alterações nos privilégios tenham efeito.

Se você deseja fornecer a um usuário específico acesso de qualquer máquina em um determinado domínio (por exemplo, `meudomínio.com`), você pode utilizar uma instrução `GRANT` como a seguir:

```
mysql> GRANT ...
    ->     ON *.*
    ->     TO myusername@'%.mydomain.com'
    ->     IDENTIFIED BY 'mypassword';
```

Para realizar a mesma coisa modificando diretamente as tabelas de permissões, faça isto:

```
mysql> INSERT INTO user VALUES ('%.meudominio, 'meunomedeusuario'
           PASSWORD('minhasenha'),...);
mysql> FLUSH PRIVILEGES;
```

4.4.6 Deletando Usuários do MySQL

```
DROP USER nome_usuario
```

Este comando foi adicionado ao MySQL 4.1.1.

Ele apaga um usuário que não possua nenhum privilágio.

Para deletar um usuário do MySQL você usar o seguinte procedimento, realizando os passos na ordem mostrada.

1. Verifique quais privilégios o usuário tem com `SHOW PRIVILEGES`. Veja Seção 4.6.8.11 [SHOW PRIVILEGES], Página 325.
2. Delete todos os privilégios do usuário com `REVOKE`. Veja Seção 4.4.1 [GRANT], Página 254.
3. Delete o usuário com `DROP USER`.

Se você estiver usando uma versão mais antiga do MySQL você deve primeiro revogar os privilégios e então deletar o usuário com:

```
DELETE FROM mysql.user WHERE user='username' and host='hostname';
FLUSH PRIVILEGES;
```

4.4.7 Limitando os Recursos dos Usuários

A partir do MySQL 4.0.2 pode se limitar certos recursos por usuários.

Até então, o único método disponível de limitação de uso do servidor MySQL era canfigurar a variável de inicialização `max_user_connections` para um valor diferente de zero. Mas este método é estritamente global e não permite o gerenciamento de usuários individuais, o que pode ser de interresse particular do Provedor de Serviços Internet.

Consequentemente, o gerenciamento de três recursos é introduzido no nível de usuário individual:

- Número de todas as consultas por hora: Todos os comandos que podiam ser executados por um usuário.
- Número de todas as atualizações por hora: Qualquer comando que altera qualquer tabela ou banco de dados.
- Númeor de conexões feitas por hora: Novas conexões abertas por hora.

Um usuário no contexto mencionado acima é uma única entrada na tabela `user`, que é identificada unicamente por suas colunas `user` e `host`.

Todos os usuários não são limitados por padrão no uso dos recursos acima, a menos que os limites sejam garantidos a eles. Estes limites podem ser concedidos **apenas** através do `GRANT (*.*)` global, usando esta sintaxe:

```
GRANT ... WITH MAX_QUERIES_PER_HOUR N1
               MAX_UPDATES_PER_HOUR N2
               MAX_CONNECTIONS_PER_HOUR N3;
```

Pode-se especificar qualquer combinação dos recursos acima. `N1`, `N2` e `N3` são inteiros e significam contagem/hora.

Se os usuários alcançam o limite de conexões dentro de uma hora, não será aceita mais nenhuma conexão até o fim desta hora. De forma parecida se o usuário alcança o limite do número de consultas ou atualizações, consultas ou atualizações adicionais serão rejeitadas até que a hora acabe. Em todos os casos, uma mensagem de erro apropriada é enviada.

Os valores atualmente usados por um usuário em particular pode ser descarregados (zerados) enviando uma instrução `GRANT` com qualquer das cláusulas acima, incluíndo uma instrução `GRANT` com os valores atuais.

Os valores atuais para todos os usuários para todos os usuários serão descarregados se os privilégios forem recarregados (no servidor ou usando `mysqladmin reload`) ou se o comando `FLUSH USER_RESOURCES` é executado.

O resurso está habilitado assim que e concedido a um único usuário qualquer das cláusulas `GRANT` de limitação.

Como um prerequisito para a habilitação deste recurso, a tabela `user` no banco de dados `mysql` deve conter as colunas adicionais, como definido no script de criação de tabelas `mysql_install_db` e `mysql_install_db.sh` no subdiretório '`scripts`'.

4.4.8 Configurando Senhas

Na maioria dos casos você deve utilizar `GRANT` para configurar seus usuários e senhas, portanto, as informações exibidas a seguir são aplicadas somentes para usuários avançados. Veja Seção 4.4.1 [`GRANT`], Página 254.

Os exemplos nas seções precedentes ilustram um princípio importante: quando você armazena uma senha não-vazia utilizando `INSERT` ou `UPDATE` você deve utilizar a função `PASSWORD()` para criptografá-la. Isto é porque a tabela `user` armazena senhas na forma criptografada, e não como texto puro. Se você esquecer deste fato, é provável que você possa tentar configurar senhas desta forma:

```
shell> mysql -u root mysql
mysql> INSERT INTO user (Host,User,Password)
       VALUES('%','jeffrey','biscuit');
mysql> FLUSH PRIVILEGES;
```

O resultado é que o valor `'biscuit'` é armazenado como a senha na tabela `user`. Quando o usuário `jeffrey` tentar se conectar ao servidor utilizando esta senha, o cliente `mysql` a criptografa utilizando a função `PASSWORD()`, gerando um vetor de autenticação baseado em uma senha **criptografada** e um número randômico, obtido do servidor, e envia o resultado ao servidor. O servidor usa o valor do campo `password` na tabela `user` (que é o valor `'biscuit'` **não criptografado**) para realizar os mesmos cálculos e comparar os resultados. A comparação falha e o servidor rejeita a conexão:

```
shell> mysql -u jeffrey -pbiscuit test
Access denied
```

As senhas devem ser criptografadas quando elas são inseridas na tabela `user`, portanto a instrução `INSERT` deveria ter sido informada no seguinte formato:

```
mysql> INSERT INTO user (Host,User,Password)
       VALUES('%','jeffrey',PASSWORD('biscuit'));
```

Você deve também utilizar a função `PASSWORD()` quando utilizar instruções `SET PASSWORD`:

```
mysql> SET PASSWORD FOR jeffrey@"%" = PASSWORD('biscuit');
```

Se você configurar senhas utilizando a instrução `GRANT ... IDENTIFIED BY` ou o comando `mysqladmin password`, a função `PASSWORD()` é desnecessária. Ambos tomam o cuidado de criptografar a senha para você, então você deve especificar a senha `'biscuit'` desta forma:

```
mysql> GRANT USAGE ON *.* TO jeffrey@"%" IDENTIFIED BY 'biscuit';
```

ou

```
shell> mysqladmin -u jeffrey password biscuit
```

NOTA: `PASSWORD()` é diferente da senha criptografada do Unix.

4.4.9 Mantendo Sua Senha Segura

Não é aconselhável especificar uma senha de uma forma que a exponha e possa ser descoberta por outros usuários. Os métodos que você pode usar para especificar sua senha quando executar programas clientes são listados abaixo, juntamente com as determinações de riscos de cada método:

- Nunca forneça a um usuário normal acesso à tabela `mysql.user`. O conhecimento de uma senha criptografada possibilita a conexão como este usuário. As senhas só estão embaralhadas para que não seja possível chegar à senha real que foi usada (acontece muito a utilização de senhas similares em outras aplicações).
- Uso da opção `-psua_senha` ou `--password=sua_senha` na linha de comando. Isto é conveniente mas inseguro, porque sua senha se torna visível para programas de informação do sistema (como no `ps`) que pode ser chamado por outros usuários para exibir linhas de comando. (clientes MySQL normalmente gravam zeros em cima do argumento da linha de comando durante sua sequência de inicialização, mas ainda existe um breve intervalo no qual o valor está visível.)
- Uso das opções `-p` ou `--pasword` (sem especificar o valor `sua_senha`). Neste caso, o programa cliente solicita a senha do terminal:

  ```
  shell> mysql -u user_name -p
  Enter password: ********
  ```

  Os caracteres '`*`' representam sua senha.

  É mais seguro digitar sua senha desta forma do que especificá-la na linha de comando porque ela não fica visível a outros usuários. Entretanto este método de digitar uma senha é válido somente para programas que você executa de forma interativa. Se você deseja chamar um cliente de um script que não execute interativamente, não existirá oportunidade de digitar a senha do terminal. Em alguns sistemas, você pode descobrir que a primeira linha do seu script é lida e interpretada (incorretamente) como sua senha!
- Armazenar a sua senha em um arquivo de configuração. Por exemplo, você pode listar sua senha na seção `[client]` do arquivo '`.my.cnf`' no seu diretório home:

  ```
  [client]
  password=sua_senha
  ```

  Se você armazenar sua senha em um arquivo '`.my.cnf`', o arquivo não pode ser lido por seu grupo ou pelos outros usuários. Tenha certeza que o modo de acesso do arquivo é `400` ou `600` Veja Seção 4.1.2 [Option files], Página 216.
- Você pode armazenar sua senha na variável de ambiente `MYSQL_PWD`, mas este método deve ser considerado extremamente inseguro e não deve ser usado. Algumas versões de `ps` incluem uma opção para exibir o ambiente de processos em execução; sua senha estaria em texto puro para a leitura para todos os usuários. Mesmo em sistemas sem

esta versão do `ps`, seria imprudência assumir que não existe outro método para observar o ambiente de processos. Veja Apêndice F [Variáveis de ambiente], Página 1092.

Em resumo, os métodos mais seguros seriam que o programa cliente solicitasse a senha ou especificar a senha em um arquivo '`.my.cnf`' corretamente protegido.

4.4.10 Usando Conexões Seguras

4.4.10.1 Conceitos Basicos

A partir da versão 4.0.0, o MySQL tem suporte a conexões cri[ptografadas com SSL. Para entender como o MySQL usa SSL, é necessário explicar alguns conceits básicos de SSL e X509. A pessoal que já estão familiarizada com eles podem saltar esta parte.

Por padrão o MySQL não usa conexões criptografadas entre o cliente e o servidor. Isto significa que qualquer um pode observar todo o tráfico e ver os dados enviados e recebidos. Podiase até mesmo alterar os dados enquanto eles estavam em transito entre o cliente e o servidor. Algumas vezes você precisao mover informações sobre redes públicas de um modo seguro; em tais casos, usar uma conexão sem criptografia é inaceitável.

SSL é um protocolo que utiliza diferentes algorítimos de criptografia para assegurar que os dados recebidos por uma rede pública são confiáveis. Ele tem um mecanismo para detectar qualquer alteração, perda ou reenvio de dados. SSL também incorpora algoritmos para reconhecer e fornecer identidades de verificação usando o padrão X509.

Criptografia é o modo de tornar qualquer tipo de dado ilegível. De fato, as práticas de hoje precisam de muitos elementos de segurança adicionais para algoritmos de criptografia. Eles devem resistir a muitos tipos de atques conhecidos como apenas alterando a ordem da mensagem criptografada ou emviando o dado duas vezes.

X509 é um padrão que torna possível identificar alguém na Internet. Ele é mais comumente usado em aplicações e-commerce. Em termos básicos, deve haver algumas empresas (chamadas "Autoridades de Certificação") que atribuem certificados eletrônicos para qualquer um que precise deles. Os certificados se baseiam em algorítmos de criptografia assimétricos que possuem duas chaves de criptografia (uma chave pública e uma chave secreta). Um proprietário de certificado pode provar a sua identidade mostrnado este certificado para outra parte. Um certificado consiste das chaves públicas do proprietário. Qualquer dados criptografado com esta chave pública pode ser descriptografada apenas usando a chave secreta correspondente, que é guardada pelo dono do certificado.

O MySQL não utiliza conexões criptografadas por padrão, porque fazendo isto tornaria o protocolo cliente/servidor muito lento. Qualquer tipo de funcionalidade adiocional exige que o conputador faça um trabalho adicional e a criptografia de dados é uma operação intensiva da CPU que exige tempo e pode atrasar o MySQL nas tarefas principais. Por padrão o MySQL é ajustado para ser o mais rápido possível.

Se você precisa de mais informações sobre SSL, X509 ou criptografia, você deve usar se mecanismo de busca favorita na Internet para procurar sobre o assunto que está interessado.

4.4.10.2 Exigências

Para conseguir conexões seguras para trabalhar com o MySQL você deve fazer o seguinte:

1. Insatale o biblioteca OpenSSL. Testamos o MySQL com OpenSSL 0.9.6. `http://www.openssl.org/`.
2. Configure o MySQL com `--with-vio --with-openssl`.
3. Se você estiver usando um instalação antiga do MySQL, você tem que atualizar a sua tabela `mysql.user` com algumas novas colunas relacionadas a SSL. Isto é necessário se suas tabelas de permissões são de uma versão anterior ao MySQL 4.0.0. O procedimento está descrito em Seção 2.5.6 [Upgrading-grant-tables], Página 130.
4. Você pode verificar se um servidor `mysqld` em execução suporta OpenSSL examinando se `SHOW VARIABLES LIKE 'have_openssl'` retorna `YES`.

4.4.10.3 Configurando Certificados SSL para o MySQL

Aqui está um exemplo para configurar certificados SSL para o MySQL:

```
DIR=‘pwd‘/openssl
PRIV=$DIR/private

mkdir $DIR $PRIV $DIR/newcerts
cp /usr/share/ssl/openssl.cnf $DIR
replace ./demoCA $DIR -- $DIR/openssl.cnf

# Crie os aarquivos necessário: $database, $serial e o diretório
$new_certs_dir (opcional)

touch $DIR/index.txt
echo "01" > $DIR/serial

#
# Geração do Certificate Authority(CA)
#

openssl req -new -x509 -keyout $PRIV/cakey.pem -out $DIR/cacert.pem \
    -config $DIR/openssl.cnf

# Saída exemplo:
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ..............++++++
# .........++++++
# writing new private key to '/home/monty/openssl/private/cakey.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
```

```
# You are about to be asked to enter information that will be incorporated
# into your certificate request.
# What you are about to enter is what is called a Distinguished Name or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL admin
# Email Address []:

#
# Create server request and key
#
openssl req -new -keyout $DIR/server-key.pem -out \
    $DIR/server-req.pem -days 3600 -config $DIR/openssl.cnf

# Saída exemplo:
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ..++++++
# ..........++++++
# writing new private key to '/home/monty/openssl/server-key.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be incorporated
# into your certificate request.
# What you are about to enter is what is called a Distinguished Name or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL server
# Email Address []:
#
# Please enter the following 'extra' attributes
# to be sent with your certificate request
```

```
# A challenge password []:
# An optional company name []:

#
# Remove the passphrase from the key (optional)
#

openssl rsa -in $DIR/server-key.pem -out $DIR/server-key.pem

#
# Assina o certificado do servidor
#
openssl ca  -policy policy_anything -out $DIR/server-cert.pem \
    -config $DIR/openssl.cnf -infiles $DIR/server-req.pem

# Saída exemplo:
# Using configuration from /home/monty/openssl/openssl.cnf
# Enter PEM pass phrase:
# Check that the request matches the signature
# Signature ok
# The Subjects Distinguished Name is as follows
# countryName           :PRINTABLE:'FI'
# organizationName      :PRINTABLE:'MySQL AB'
# commonName            :PRINTABLE:'MySQL admin'
# Certificate is to be certified until Sep 13 14:22:46 2003 GMT (365 days)
# Sign the certificate? [y/n]:y
#
#
# 1 out of 1 certificate requests certified, commit? [y/n]y
# Write out database with 1 new entries
# Data Base Updated

#
# Create client request and key
#
openssl req -new -keyout $DIR/client-key.pem -out \
    $DIR/client-req.pem -days 3600 -config $DIR/openssl.cnf

# Saída exemplo:
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ..................................++++++
# ..........................................++++++
# writing new private key to '/home/monty/openssl/client-key.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
```

```
# You are about to be asked to enter information that will be incorporated
# into your certificate request.
# What you are about to enter is what is called a Distinguished Name or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL user
# Email Address []:
#
# Please enter the following 'extra' attributes
# to be sent with your certificate request
# A challenge password []:
# An optional company name []:

#
# Remove a passphrase from the key (optional)
#
openssl rsa -in $DIR/client-key.pem -out $DIR/client-key.pem

#
# Sign client cert
#

openssl ca  -policy policy_anything -out $DIR/client-cert.pem \
    -config $DIR/openssl.cnf -infiles $DIR/client-req.pem

# Saída exemplo:
# Using configuration from /home/monty/openssl/openssl.cnf
# Enter PEM pass phrase:
# Check that the request matches the signature
# Signature ok
# The Subjects Distinguished Name is as follows
# countryName           :PRINTABLE:'FI'
# organizationName      :PRINTABLE:'MySQL AB'
# commonName            :PRINTABLE:'MySQL user'
# Certificate is to be certified until Sep 13 16:45:17 2003 GMT (365 days)
# Sign the certificate? [y/n]:y
#
#
# 1 out of 1 certificate requests certified, commit? [y/n]y
# Write out database with 1 new entries
```

```
# Data Base Updated

#
# Create a my.cnf file that you can use to test the certificates
#

cnf=""
cnf="$cnf [client]"
cnf="$cnf ssl-ca=$DIR/cacert.pem"
cnf="$cnf ssl-cert=$DIR/client-cert.pem"
cnf="$cnf ssl-key=$DIR/client-key.pem"
cnf="$cnf [mysqld]"
cnf="$cnf ssl-ca=$DIR/cacert.pem"
cnf="$cnf ssl-cert=$DIR/server-cert.pem"
cnf="$cnf ssl-key=$DIR/server-key.pem"
echo $cnf | replace " " '
' > $DIR/my.cnf

#
# To test MySQL

mysqld --defaults-file=$DIR/my.cnf &

mysql --defaults-file=$DIR/my.cnf
```

Você também pode testar sua configuração modificando o arquivo '`my.cnf`' acima para fazer referência aos certificados de demonstração no diretório mysql-dist-fonte/SSL.

4.4.10.4 Opções SSL do `GRANT`

O MySQL pode verificar atributos do certificado X509 em adição ao esquema normal de usuário/senha. Todas as opções comuns ainda são exigidas (usuário, senha, máscara do endereço IP, noome tabela/banco de dados).

Existem diferentes possibilidades para limitarmos as conexões:

- Sem nenhuma opção SSL ou X509, todos os tipos de conexões criptografadas/ descriptografadas são permitidas se o usuário e senha são válidos.
- A opção `REQUIRE SSL` limita o servidor para permitir apenas conexões criptografadas SSL. Note que esta opção pode ser omitida se não houver nenhum registro ACL que permita conexões não SSL.

  ```
  mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
      -> IDENTIFIED BY 'goodsecret' REQUIRE SSL;
  ```

- `REQUIRE X509` significa que o cliente deve ter um certificado válido mas não nos procupamos sobre o certificado, o emissor ou assunto exato. A única restrição é que deve ser possível verificar a sua assinatura com um dos certificados CA.

  ```
  mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
  ```

```
    -> IDENTIFIED BY 'goodsecret' REQUIRE X509;
```

- `REQUIRE ISSUER 'emissor'` coloca uma restrição na tentativa de conexão: O cliente deve apresentar um certificado X509 válido emitido pelo CA `'emissor'`. Usar o certificado X509 sempre implica em criptografia, assim a opção `SSL` é desnecessária.

  ```
  mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
      -> IDENTIFIED BY 'goodsecret'
      -> REQUIRE ISSUER 'C=FI, ST=Some-State, L=Helsinki,
      '> O=MySQL Finland AB, CN=Tonu Samuel/Email=tonu@mysql.com';
  ```

- `REQUIRE SUBJECT 'assunto'` exige que o cliente tenha um certificado X509 com o assunto `'assunto'`. Se o cliente apresenta um certificado que é valido mas tem um `'assunto'` diferente, a conexão é disabilitada.

  ```
  mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
      -> IDENTIFIED BY 'goodsecret'
      -> REQUIRE SUBJECT 'C=EE, ST=Some-State, L=Tallinn,
      '> O=MySQL demo client certificate,
      '> CN=Tonu Samuel/Email=tonu@mysql.com';
  ```

- `REQUIRE CIPHER 'método'` é necessário para assegurar que uma criptografia forte será usada. O SSL pode ser fraco se algoritmos antigos com chaves de criptografias curtas são usados. Usando esta opção, podemos pedir por algum método de criptografia exato para permitir a conexão.

  ```
  mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
      -> IDENTIFIED BY 'goodsecret'
      -> REQUIRE CIPHER 'EDH-RSA-DES-CBC3-SHA';
  ```

  As opções `SUBJECT`, `ISSUER` e `CIPHER` podem ser combinadas na cláusula `REQUIRE` desta forma:

  ```
  mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
      -> IDENTIFIED BY 'goodsecret'
      -> REQUIRE SUBJECT 'C=EE, ST=Some-State, L=Tallinn,
      '> O=MySQL demo client certificate,
      '> CN=Tonu Samuel/Email=tonu@mysql.com'
      -> AND ISSUER 'C=FI, ST=Some-State, L=Helsinki,
      '> O=MySQL Finland AB, CN=Tonu Samuel/Email=tonu@mysql.com'
      -> AND CIPHER 'EDH-RSA-DES-CBC3-SHA';
  ```

  A partir do MySQL 4.0 a palavra chave `AND` é opcional entre opções `REQUIRE`.

  A ordem das opções não importa, mas nenhuma opção pode ser especificada duas vezes.

4.4.10.5 Opções SSL de Linha de Comando

A seguinte tabela lista opções que são usadas para especificar o uso de SSL, arquivos de certificado e arquivos de chaves. Estas opções estão disponíveis a partir do MySQL 4.0. Elas podem ser dadas na linha de comando ou no arquivo de opção.

`--ssl` Para o servidor, especifica que o servidor permite conexões SSL. Para um programa cliente, permite que o cliente se conecte ao servidor usando SSL. Esta

opção por si só não é suficiente para fazer uma conexão SSL ser usada. Você também deve especificar as opções `--ssl-ca`, `--ssl-cert`, e `--ssl-key`.

Note que esta opção não **exige** uma conexão SSL. Por exemplo, se o servidor ou clienteestá compilado sem suporte SSL, uma conexão não criptografada normal será usada.

O modo seguro de de se certificar que uma conexão SSL será usada é criar uma conta no servidor que inclua uma cláusula `REQUIRE SSL` na instrução `GRANT`. Então use esta conta para se conectar ao servidor, com um servidor e cliente que tenham suporte a SSL habilitado.

Você pode usar esta opção para indicar que a conexão não deve usar SSL. Faça isto especificando a opção como `--skip-ssl` ou `--ssl=0`.

`--ssl-ca=file_name`
O caminho para um arquivo vom uma lista de Certifcados SSL confiáveis.

`--ssl-capath=directory_name`
O caminho para um diretório que contém certificados SSL confiáveis no formato pem.

`--ssl-cert=file_name`
O nome do arquivo de certificado SSL usado para estabelecer uma conexão segura.

`--ssl-cipher=cipher_list`
Uma lista de chaves permitidas, usado para criptografia SSL. `cipher_list` tem o mesmo formato que o comando `openssl ciphers`.

Example: `--ssl-cipher=ALL:-AES:-EXP`

`--ssl-key=file_name`
O nome do arquivo de chave SSL a ser usado para estabelecer uma conexão segura.

4.5 Prevenção de Disastres e Recuperação

4.5.1 Backups dos Bancos de Dados

Como as tabelas do MySQL são armazenadas como arquivos, é mais fácil realizar um backup. Para obter um backup consistente, faça um `LOCK TABLES` nas tabelas relevantes seguido por `FLUSH TABLES` para as tabelas. Veja Seção 6.7.5 [`LOCK TABLES`], Página 617. Veja Seção 4.6.4 [`FLUSH`], Página 299. Você só precisa de um bloqueio de leitura; isto possibilita outras threads a continuarem a pesquisar nas tabelas enquanto você copia os arquivos no diretório do banco de dados. O `FLUSH TABLE` é necessário para garantir que todas as páginas ativas de índices serão escritas em disco antes de iniciar o backup.

A partir das versões 3.23.56 e 4.0.12 `BACKUP TABLE` não permitirá que você sobrescreva arquivos exixtentes já que isso colocaria em risco a segurança.

Se você desejar realizar um backup ao nível da linguagem SQL de um tabela, você pode utilizar `SELECT INTO OUTFILE` ou `BACKUP TABLE`. Veja Seção 6.4.1 [SELECT], Página 562.Veja Seção 4.5.2 [BACKUP TABLE], Página 277.

Outra maneira de efetuar um backup de um banco de dados é utilizar o programa `mysqldump` ou o `script mysqlhotcopy`. Veja Seção 4.9.7 [mysqldump], Página 361. Veja Seção 4.9.8 [mysqlhotcopy], Página 366.

1. Fazer um backup completo do seu banco de dados:

   ```
   shell> mysqldump --tab=/path/to/some/dir --opt db_name

   ou

   shell> mysqlhotcopy db_name /path/to/some/dir
   ```

   Você também pode simplesmente copiar os arquivos das tabelas ('`*.frm`', '`*.MYD`') e os arquivos '`*.MYI`') quando o servidor não estiver atualizando nada. O script `mysqlhotcopy` utiliza este método. (Mas nopte que estes métodos não funcionarão se seu banco de dados contém tabelas `InnoDB`. `InnoDB` não armazena o conteúdo das tabelas em diretórios de banco de dados, e o `mysqlhotcopy` funciona apenas para tabelas `MyISAM` e `ISAM`.)

2. Interrompa o `mysqld` caso ele esteja em execução, depois inicie-o com a opção `--log-bin[=nome_arquivo]`. Veja Seção 4.10.4 [Log binário], Página 374. Os arquivos de log binário fornecem a informação necessária para replicar alterações ao banco de dados que forem feitas depois do ponto em que você executou `mysqldump`.

Se o seu servidor MySQL é um slave, seja qual for o método de backup que você escolha, quando você faz backup dos dados do slave, você deve também fazer backup dos arquivos '`master.info`' e '`relay-log.info`' que são necessários para continuar a replicação depois que você restaurar os dados do slave. Se seu slave está sujeito a replicação de comandos `LOAD DATA INFILE`, você também deve fazer backup dos arquivos '`SQL_LOAD-*`' que podem existir no diretório especificado pela opção '`slave-load-tmpdir`'. (A localização padrão desta opção é o valor da variável `tmpdir`se não especificado.) O slave precisará destes arquivos para continuar a replicação de qualquer `LOAD DATA INFILE` interrompido.

Se você necessita restaurar alguma coisa, tente primeiro recuperar suas tabelas utilizando `REPAIR TABLE` ou `myisamchk -r`. Isto deve funcionar em 99.9% de todos os caso, Se o `myisamchk` falhar, tente o seguinte procedimento: (Isto só irá funcionar se você iniciou o MySQL com `--log-update`, veja Seção 4.10.4 [Binary log], Página 374,):

1. Restaure o backup original feito com o `mysqldump` ou backup binário.
2. Execute o seguinte comando para re-executar as atualizações armazenadas no log binário:

   ```
   shell> mysqlbinlog hostname-bin.[0-9]* | mysql
   ```

   Em seu caso você pode querer re-executar apenas alguns log binários, a partir de certas posiçõs (normalmente você quer re-executar todos os log binários a partir da data de restauração do backup, co exceção de algumas consultas erradas). Veja Seção 4.9.5 [mysqlbinlog], Página 358 fpara mais informações sobre o utilitário `mysqlbinlog` e como usá-lo.

   Se você estiver utilizando o log atualizado, você pode executar o conteúdo do log de atualização desta forma:

   ```
   shell> ls -1 -t -r hostname.[0-9]* | xargs cat | mysql
   ```

O comando `ls` é usado para obter todos os arquivos de log na ordem correta.

Você pode também fazer backups seletivos com `SELECT * INTO OUTFILE 'nome_arquivo' FROM nome_tabela` e restaurar com `LOAD DATA INFILE 'nome_arquivo' REPLACE...`. Para evitar registros duplicados, você precisará de um chave `PRIMARY KEY` ou uma `UNIQUE` na tabela. A palavra chave `REPLACE` substitui os antigos registros com os novos quando um novo registro duplica um antigo registro em uma chave de valores únicos.

Se você tiver problemas de performance realizando backups no seu sistema, você pode resolver isto configurando uma replicação e fazendo os backups na máquina slave no lugar da master. Veja Seção 4.11.1 [Introdução a Replicação], Página 378.

Se você estiver utilizando um sistema de arquivos Veritas, você pode fazer:

1. Executar em um cliente (perl ?) `FLUSH TABLES WITH READ LOCK`
2. Bifurcar uma shell ou executar em outro cliente `mount vfxs snapshot`.
3. Executar no primeiro cliente `UNLOCK TABLES`
4. Copiar arquivos do snapshot
5. Desmontar snapshot

4.5.2 Sintaxe de `BACKUP TABLE`

```
BACKUP TABLE nome_tabela[,nome_tabela...] TO '/caminho/para/diretório/backup'
```

Faz uma cópia de todos os arquivos de tabela para o diretório de backup que é o mínimo necessário para restaurá-lo. Atualmente só funciona para tabelas `MyISAM`. Para tabela `MyISAM`, copia os arquivos `.frm` (definições) e `.MYD` (dados). O arquivo de índice pode ser reconstruído a partir destes dois.

Antes de utilizar este comando, por favor veja Veja Seção 4.5.1 [Backup], Página 275.

Durante o backup, o bloqueio de leitura (read lock) será usado para cada tabela, uma de cada vez, à medida que o backup é realizado. Se você deseja fazer backup de diversas tabelas como um snapshot, você deve primeiro usar `LOCK TABLES` obtendo um bloqueio de leitura para cada tabela no grupo.

O comando retorna uma tabela com as seguintes colunas:

Coluna	**Valor**
Table	Nome da Tabela
Op	Sempre `backup`
Msg_type	Um dos seguintes: `status`, `error`, `info` ou `warning`.
Msg_text	A mensagem

Note que o comando `BACKUP TABLE` está disponível somente no MySQL versão 3.23.25 e posterior.

4.5.3 Sintaxe de `RESTORE TABLE`

```
RESTORE TABLE nome_tabela[,nome_tabela...] FROM '/caminho/para/diretório/backup'
```

Restaura a tabela ou tabelas utilizando o backup feito com `BACKUP TABLE`. Tabelas existentes não serão reescritas - se você tentar restaurar sobre uma tabela existente, obterá um erro. A restauração demora mais tempo do que o backup pois é necessário reconstruir o

índice. Quanto mais chaves tiver, mais demorado será. Como no comando `BACKUP TABLE`, atualmente só funciona com tabelas `MyISAM`.

O comando retorna uma tabela com as seguintes colunas:

Coluna	**Valor**
Table	Nome da Tabela
Op	Sempre `restore`
Msg_type	Um dos seguintes: `status`, `error`, `info` ou `warning`
Msg_text	A mensagem

4.5.4 Sintaxe de `CHECK TABLE`

```
CHECK TABLE nome_tabela[,nome_tabela...] [opção [opção...]]

opção = QUICK | FAST | MEDIUM | EXTENDED | CHANGED
```

`CHECK TABLE` funciona somente em tabelas `MyISAM`. Em tabelas `MyISAM` é a mesma coisa que executar `myisamchk --medium-check nome_tabela` na tabela.

Se você não especificar nenhuma opção, `MEDIUM` é usado.

Verifica se existem erros na(s) tabela(s). Para as tabelas `MyISAM` as estatísticas das chaves são atualizadas. O comando retorna uma tabela com as seguintes colunas:

Coluna	**Valor**
Table	Nome da Tabela.
Op	Sempre `check`
Msg_type	Um dos seguintes: `status`, `error`, `info`, or `warning`
Msg_text	A mensagem

Note que a instrução pode produzir várias linhas de informações para cada tabela conferida. A última linha irá ser do tipo `Msg_type status` e normalmente deve estar `OK`. Se você não obteve `OK` ou `Not checked`, deve ser executado, normalmente, um reparo da tabela. Veja Seção 4.5.6 [Table maintenance], Página 280. `Table is already up to date` significa que o gerenciador de armazenamento para a tabela indica que não há necessidade de verificar a tabela.

Os diferentes tipos de consistências são as seguintes:

Tipo	**Significado**
`QUICK`	Não busca os registros verificando ligações incorretas.
`FAST`	Só confere tabelas que não foram fechadas corretamente.
`CHANGED`	Só verifica as tabelas que foram alteradas desde a última conferência ou que não foram fechadas corretamente.
`MEDIUM`	Busca os registros para verificanado que ligações removidas estão ok. Isto também calcula uma chave de conferência para os registros e verifica isto com um checksum calculado para as chaves.
`EXTENDED`	Faz uma busca completa nas chaves para todas as chaves em cada registro. Isto assegura que a tabela está 100% consistente, mas pode demorar muito tempo para executar!

Para tabelas MyISAM de tamanho dinâmico, uma verificação iniciada sempre fará uma verificação MEDIUM. Para registros de tamanho estático nós saltamos a busca de registros para QUICK e FAST já que os registros estão raramente corrompidos.

Você pode combinar opções de consistência como no exemplo a seguir que faz uma verificação rápida na tabela para ve se ela foi fechada corretamente:

```
CHECK TABLE test_table FAST QUICK;
```

NOTA: em alguns casos CHECK TABLE irá alterar a tabela! Isto acontece se a tabela estiver marcada como 'corrupted' (corrompida) ou 'not closed properly' (não foi fechada corretamente) mas o CHECK TABLE não encontrar não encontrar nenhum problema na tabela. Neste caso, CHECK TABLE irá marcar a tabela como ok.

Se uma tabela estiver corrompida, é preferível que seja um problema nos índices e não na parte de dados. Todos os tipos de consistência acima sempre confere os índices e deve então encontrar a maioria dos erros.

Se você só quiser conferir uma tabela que acredita estar ok, você não deve utilizar nenhuma opção para o comando check ou utilizar a opção QUICK. O último deve ser utilizado quando você estiver com pressa e o risco do QUICK não encontrar um erro no arquivo de dados for mínimo (Na maioria dos casos o MySQL pode encontrar, sob utilização normal, qualquer erro no arquivo de dados. Se isto ocorrer, então a tabela será marcada como 'corrupted', neste caso a tabela não poderá ser utilizada até ser reparada).

FAST e CHANGED são normalmente chamados a partir de um script (um exemplo é ser executado a partir do cron) Se você desejar conferir suas tabelas de tempos em tempos. Na maioria dos casos, o FAT é uma opção melhor que CHANGED. (O único caso em que isto não acontece é quando você suspeita que encontrou um bug no código do MyISAM.).

EXTENDED deve ser utilizado somente depois de ter executado um check normalmente, mas continuar obtendo erros de uma tabela quando o MySQL tenta atualizar um registro ou encontrar um registro pela chave (isto seria muito difícil ocorrer caso uma conferência normal tenha executado com sucesso!).

Alguns problemas relatados por CHECK TABLE, não podem ser corrigidas automaticamente:

- `Found row where the auto_increment column has the value 0.`

  Isto significa que você possui um registro na tabela onde o campo índice que utiliza o recurso auto_increment contem o valor 0. (É possível criar um registro onde a coluna de auto incremento seja 0 definindo explicitamente 0 em uma instrução UPDATE).

  Isto não é exatamente um erro, mas pode causar problemas se você decidir descarregar a tabela e restaurá-la ou executar um ALTER TABLE na tabela. Neste caso a coluna de auto incremento irá alterar seu valor, de acordo com as regras das colunas de auto incremento, que pode causar problemas como um erro de chave duplicada.

  Para se livrar do alerta, basta executar uma instrução UPDATE para configurar a coluna para algum outro valor diferente de 0.

4.5.5 Sintaxe do REPAIR TABLE

```
REPAIR [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name[,tbl_name...] [QUICK] [EXTENDED]
```

REPAIR TABLE funciona somente em tabelas MyISAM e é a mesma coisa que executar myisamchk -r nome_tabela na tabela.

Normalmente você nunca deve executar este comando, mas se um disastre ocorrer você vai precisar recuperar seus dados de uma tabela MyISAM utilizando `REPAIR TABLE`. Se as suas tabelas estiverem muito corrompidas, você deve encontrar a razão, para eleiminar a necessidade de se usar `REPAIR TABLE`! Veja Seção A.4.1 [Crashing], Página 925. Veja Seção 7.1.3 [MyISAM table problems], Página 636.

`REPAIR TABLE` repara uma tabela possivelmente corrompida. O comando retorna uma tabela com as seguintes colunas:

Coluna	**Valor**
Table	Nome da Tabela
Op	Sempre `repair`
Msg_type	Um dos seguintes: `status`, `error`, `info` ou `warning`
Msg_text	A mensagem

Note que a instrução pode produzir várias linhas de informações para cada tabela recuperada. A ultima linha será de `Msg_type status` e normalmente deve exibir `OK`. Se o retorno não for `OK`, você pode tentar reparar a tabela com `myisamchk -o`, já que `REPAIR TABLE` ainda não implementa todas as opções de `myisamchk`. Futuramente iremos torná-lo mais flexível.

Se o parâmetro `QUICK` for especificado, `REPAIR` tenta reparar somente a árvore de índices.

Se você utilizar `EXTENTED`, o MySQL criará o índice, registro a registro em vez de criar um índice de uma vez com ordenação; Isto pode ser melhor que a ordenação em chaves de tamanho fixo se você tiver grandes chaves do tipo `char()` que compactam muito bem.

No `MySQL` 4.0.2, existe um modo `USE_FRM` para `REPAIR`. Use-o se o arquivo '`.MYI`' estiver faltando ou o seu cabeçalho estiver corrompido. Neste modo o MySQL recriará a tabela, usando a informação do arquivo '`.frm`'. Este tipo de reparo não pode ser feito com `myisamchk`.

Aviso: Se o `mysqld` morre durante um `REPAIR TABLE`, é essencial que você faça imediatamente outro `REPAIR` na tabela antes de executar qualquer outro comando nela. (Claro que é sempre bom inciar com um backup). No pior caso você pode ter um novo arquivo de índice limpo sem informação sobre o arquivo de dados e quando você executar o próximo comando o arquivo de dados pode ser sobreescrito. Isto não é um cenário desejável, mas possível.

Antes do MySQL 4.1.1, o comando `REPAIR` não era gravado no log binário. Desde o MySQL 4.1.1. eles são escritos no log binário a menos que a palavra chave opcional `NO_WRITE_TO_BINLOG` (ou seu alias `LOCAL`) seja usada.

4.5.6 Utilizando `myisamchk` para Manutenção de Tabelas e Recuperação em Caso de Falhas

A partir do MySQL versão 3.23.13 você pode mandar verificar as tabelas MyISAM com o comando `CHECK TABLE`. Veja Seção 4.5.4 [CHECK TABLE], Página 278. Pode-se reparar tabelas com o comando `REPAIR TABLE`. Veja Seção 4.5.5 [REPAIR TABLE], Página 279.

Para verificar/reparar tabelas MyISAM (`.MYI` e `.MYD`) você deve utilizar o utilitário `myisamchk`. Para consistir/reparar tabelas ISAM (`.ISM` e `.ISD`) você deve usar o utilitário `isamchk`. Veja Cap“ptexi tulo 7 [Tipos de Tabelas], Página 630.

No texto a seguir iremos comentar sobre o `myisamchk`, mas tudo também se aplica ao antigo `isamchk`.

Você pode utilizar o utilitário `myisamchk` para obter informações sobre suas tabelas de bancos de dados, verficá-las, repará-las ou otimizá-las. As seguintes seções descrevem como executar `myisamchk` (incluindo uma descrição de suas opções), como montar um calendário de manutenção, e como utilizar o `myisamchk` para executar suas várias funções.

Você pode, na maioria dos casos, utilizar o comando `OPTIMIZE TABLES` para otimizar e reparar tabelas, mas não é tão rápido e confiável (no caso real de erros fatais) como o `mysisamchk`. Por outro lado, `OPTIMIZE TABLE` é mais fácil de usar e você não tem que se preocupar com a recarrega das tabelas. Veja Seção 4.6.1 [`OPTIMIZE TABLE`], Página 298.

Embora os reparos realizados pelo `myisamchk` sejam bastante seguros, porém é sempre uma boa idéia fazer um backup dos dados ANTES de realizar um reparo (ou qualquer coisa que fará grandes alterações em alguma tabela)

4.5.6.1 Sintaxe do `myisamchk`

`myisamchk` é chamado desta forma:

```
shell> myisamchk [opções] nome_tabela
```

As `opções` especificam o que você deseja que o `myisamchk` faça. Elas são descritas abaixo. (Você também pode obter a lista das opções com `myisamchk --help`.) Sem opções, o `myisamchk` simplesmente checa sua tabela. Para obter maiores informações ou dizer ao `myisamchk` para tomar ações corretivas, especifique as opções descritas abaixo e nas seções seguintes.

`nome_tabela` é o nome da tabela do banco de dados que você deseja verificar/reparar. Se você executar o `myisamchk` em algum lugar diferente do diretório do banco de dados, você deve especificar o caminho para o arquivo, porque `myisamchk` não faz idéia de onde seu banco de dados se encontra. Na verdade, `myisamchk` não se importa se os arquivos estão localizados em um diretório de banco de dado; você pode copiar os arquivos que correspondem a uma tabela de banco de dados em outra localização e realizar neste outro lugar as operações corretivas.

Você pode nomear várias tabelas na linha de comando do `myisamchk` se você desejar. Você também pode especificar um nome como um arquivo de índice (com o sufixo '`.MYI`'), que lhe permite especificar todas tabelas em um diretório utilizando o padrão '`*.MYI`'. Por exemplo, se você está em um diretório de banco de dados, você pode checar todas as tabelas no diretório desta forma:

```
shell> myisamchk *.MYI
```

Se você não estiver no diretório do banco de dados, você pode verificar todas as tabelas existentes especificando o caminho para o diretório:

```
shell> myisamchk /caminho/para/banco_de_dados/*.MYI
```

Você pode verificar todas as tabelas em todos os bancos de dados especificando um meta caracter com o caminho para o diretório de banco de dados do MySQL:

```
shell> myisamchk /caminho/para/diretório_dados/*/*.MYI
```

A maneira recomendada para conferir todas as tabelas rapidamente é:

```
myisamchk --silent --fast /caminho/para/diretório_dados/*/*.MYI
isamchk --silent /caminho/para/diretório_dados/*/*.ISM
```

Se você quiser conferir todas as tabelas e reparar todas que estiverem corrompidas, pode utilizar linha a seguir:

```
myisamchk --silent --force --fast --update-state -O key_buffer=64M \
          -O sort_buffer=64M -O read_buffer=1M -O write_buffer=1M \
          /caminho/para/diretório_dados/*/*.MYI
isamchk --silent --force -O key_buffer=64M -O sort_buffer=64M \
        -O read_buffer=1M -O write_buffer=1M /caminho/para/diretório_dados/*/*.ISM
```

A linha acima assume que você tem mais de 64 MB de memória livre.

Perceba que se você obter um erro do tipo:

```
myisamchk: warning: 1 clients is using or hasn't closed the table properly
```

Isto significa que você está tentando verificar uma tabela que está sendo atualizada por outro programa (como o servidor `mysqld`) que ainda não fechou o arquivo ou que finalizou sem fechar o arquivo corretamente.

Se o `mysqld` está em execução, você deve forçar o sincronimo e fechamento de todas tabelas com `FLUSH TABLES` e assegurar que ninguém mais esteja utilizando as tabelas quando for executar o `myisamchk`. No MySQL versão 3.23 a forma mais simples de evitar este problema é utilizar `CHECK TABLE` no lugar de `myisamchk` para verificar as tabelas.

4.5.6.2 Opções Gerais do `myisamchk`

`myisamchk` suporta as seguintes opções.

`-# ou --debug=debug_options`
: Saída do log de depuração. A string `debug_options` geralmente é `'d:t:o,nomearquivo'`.

`-? ou --help`
: Exibe uma mensagem de ajuda e sai.

`-O nome=opção, --set-variable=nome=opção`
: Configura o valor de uma variável. Por favor note que as sintaxes `--set-variable=nome=valor` e `-O name=value` estão obsoletas desde o MySQL 4.0. Use `--nome=valor`. As variáveis possíveis e seus valores padrões para o myisamchk podem ser examinados com `myisamchk --help`

Variável	Valor
key_buffer_size	523264
read_buffer_size	262136
write_buffer_size	262136
sort_buffer_size	2097144
sort_key_blocks	16
decode_bits	9

`sort_buffer_size` é utilizado quando as chaves são reparadas pela ordenação das chaves, que é o caso normal quando você utiliza `--recover`.

`key_buffer_size` é utilizando quando você estiver conferindo a tabela com `--extended-check` ou quando as chaves são reparadas inserindo-as registro a

registro na tabela (como com inserts normais). O reparo através de buffer de chaves (key buffer) é utilizado nos seguintes casos:

- Se você utilizar `--safe-recover`.
- Se os arquivos temporários necessários para ordenar as chaves forem maior que o dobro do tamanho de quando se criasse o arquivo de chaves diretamente. Isto é o caso quando se tem chaves `CHAR`, `VARCHAR` ou `TEXT` tao grandes quanto necessário pela ordenação para armazenar todas as chaves durante o processo. Se você tiver muito espaço temporário e puder forçar o `myisamchk` a reparar por ordenação você pode utilizar a opção `--sort-recover`.

Reparação através do buffer de chaves (key buffer) economiza muito mais espaço em disco do que utilizando ordenação, mas é muito mais lenta.

Se você deseja uma reparação mais rápida, configure as variáveis acima para cerca de 1/4 da sua memória disponível. Você pode configurar as variáveis para valores altos, pois somente um dos buffers acima será utilizado a cada vez.

`-s ou --silent`
: Modo discreto ou silencioso. Escreve a saída somente quando um erro ocorre. Você pode utilizar `-s` duas vezes (`-ss`) para deixar o `mysisamchk` mais silencioso.

`-v ou --verbose`
: Modo prolixo. Gera mais informação de saída. Ele pode ser utilizado com `-d` e `-e`. Utilize `-v` múltiplas vezes `-vv`, `-vvv`) para gerar mais saída!

`-V ou --version`
: Exibe a versão do `myisamchk` e sai.

`-w ou, --wait`
: No lugar de gerar um erro se a tabela estiver bloqueada, espere até que a tabela fique livre antes de continuar. Perceba que se você estiver utilizando `mysqld` na tabela com `--skip-external-locking`, a tabela só pode ser trancada por outro comadno `myisamchk`.

4.5.6.3 Opções de Verificação do `myisamchk`

`-c ou --check`
: Confere por erros na tabela. Esta é a operação padrão se você não estiver utilizando opções que a anulam.

`-e ou --extend-check`
: Verifica a tabela de forma completa (que é bastante lento se você tiver vários índices). Esta opção deve ser usada somente em casos extremos. Normalmente, `myisamchk` ou `myisamchk --medium-check` deve, na maioria dos casos, estar apto a encontrar quaisquer erros na tabela.

  Se você estiver utilizando `--extended-check` e tiver muita memória, você deve aumentar um pouco o valor de `key_buffer_size`!

`-F ou --fast`
: Verifica apenas tabelas que não foram fechadas corretamente.

`-C ou --check-only-changed`
: Verifica apenas tabelas que foram alteradas desde a última verificação.

`-f ou --force`
: Reinicia o `myisamchk` com `-r` (reparos) na tabela, se `myisamchk` encontrar quaisquer erros na tabela.

`-i ou --information`
: Exibe informações e estatísticas sobre a tabela que estiver sendo verificada.

`-m ou --medium-check`
: Mais rápido que extended-check, mas encontra somente 99.99% de todos os erros. Deve, entretando, ser bom o bastante para a maioria dos casos.

`-U ou --update-state`
: Armazena no arquivo '`.MYI`' quando a tabela foi verificada e se a tabela falhou. Isto deve ser utilizado para obter o benefício integral da opção `--check-only-changed`, mas você não deve utilizar esta opção se o servidor `mysqld` esta usando a tabela e o `mysqld` esta sendo executado com `--skip-external-locking`.

`-T ou --read-only`
: Não marca as tabelas como verificadas. Isto é útil se você utiliza o `myisamchk` para verificar uma tabela que esteja em uso por alguma outra aplicação que não utiliza bloqueios (como no `mysqld --skip-external-locking`).

4.5.6.4 Opções de Reparos do `myisamchk`

As seguintes opções são usadas se você iniciar o `myisamchk` com `-r` ou `-o`:

`-B or --backup`
: Faz um backup dos arquivos '`.MYD`' como '`filename-time.BAK`'

`--correct-checksum`
: Correct checksum information for table.

`-D # ou --data-file-length=#`
: Tamanho máximo do arquivo de dados (ao recriar arquivos de dados quando eles estão 'cheios').

`-e ou --extend-check`
: Tenta recuperar todos registros possíveis do arquivo de dados. Normalmente isto irá encontrar também várias linhas com lixo. Não utiliza esta opção a menos que esteja em desespero total.

`-f ou --force`
: Sobrescreve antigos arquivos temporários ('`nome_tabela,TMD`') em vez de abortar.

`-k # ou --keys-used=#`
: Se você estiver utilizando `ISAM`, diz ao manipulador de tabelas do `ISAM` para atualizar somente os primeiros `#` índices. Se você estiver utilizando `MyISAM`,

informa quais chaves usar, onde cada bit seleciona uma chave (a primeira chave possui o bit 0). Isto pode ser utilizado para inserções mais rápidas! Índices desativados podem ser reativados utilizando `myisamchk -r`.

`-l ou --no-symlinks`
: Não segue links simbólicos. Normalmente o `myisamchk` repara a tabela para qual um link simbólico aponta. Esta opção não existe no MySQL 4.0 pois o MySQL 4.0 não irá remover links simbólicos durante os reparos.

`-p or --parallel-recover`
: Usa a mesma técnica que `-r` e `-n`, mas cria todas as chaves em paralelo, em threads diferentes. A opção foi adicionada no MySQL 4.0.2. **Este código é alfa. Use por sua conta e risco!**

`-r ou --recover`
: Pode concertar quase tudo excetos chaves únicas que não são únicas (Que é um erro extremamente indesejável com tabelas `ISAM`/`MyISAM`). Se você deseja recuperar uma tabela, esta é primeira opção a ser tentada. Somente se o `myisamchk` relatar que a tabela não pode ser recuperada pelo `-r` você deve tentar então a opção `-o`. (Perceba que no caso indesejável de `-r` falhar, o arquivo de dados continuará intacto.) Se você possui muita memória, você deve aumentar o tamanho de `sort_buffer_size`!

`-o ou --safe-recover`
: Utiliza um antigo método de recuperação (le através de todos registros na ordem e atualiza todas as árvores de índices baseado nos registros encontrados); esta opção é muito mais lenta que `-r`, mas pode tratar vários casos indesejáveis que o `-r` não consegue tratar. Este método de recuperação também utiliza muito menos espaço em disco que `-r`. Normalmente sempre se deve tentar, primeiro, um reparo com `-r`, e somente se ele falhar, usar `-o`.

  Se você possuir muita memória, você deve aumentar o tamanho de `sort_buffer_size`!

`-n ou --sort-recover`
: Força o uso de ordenação do `myisamchk` para resolver as chaves mesmo se os arquivos temporários forem muito grandes.

`--character-sets-dir=...`
: Diretório onde conjuntos de caracteres são armazenados.

`--set-character-set=name`
: Altere o conjunto de caracteres usado pelo índice

`.t ou --tmpdir=path`
: Caminho para armazenar arquivos temporários. Se isto não for configurado, `myisamchk` irá usar a variável de ambiente `TMPDIR` para isto. A partir do MySQL 4.1, `tmpdir` pode ser configurado com uma lista de caminhos separados por dois pontos `:` (ponto e virgula `;` no Windows). Eles serão usado da forma robin-round.

`-q ou --quick`
: Reparo rápido sem modificar o arquivo de dados. Pode ser fornecido um segundo `-q` para forçar o `myisamchk` para modificar o arquivo de dados original no caso de chaves duplicadas.

`-u ou --unpack`
: Descompacta arquivo empacotado com o myisampack.

4.5.6.5 Outras Opções do `myisamchk`

Outras ações que o `myisamchk` pode fazer, alem de reparar e verificar tabelas:

`-a or --analyze`
: Analiza a distribuição das chaves. Isto aumenta o desempenho de join habilitando o otimizador de joins para melhor escolher em qual ordem ele deve unir as tabelas e quais chaves ele deve usar: `myisamchk --describe --verbose table_name'` ou usar `SHOW KEYS` no MySQL.

`-d or --description`
: Exibe alguma informação sobre tabela.

`-A or --set-auto-increment[=value]`
: Força que `AUTO_INCREMENT` com um valor maior ou igual a este. Se nenhum valor é dado, então define o próximo valor `AUTO_INCREMENT` com o maior valor usado para a chave automatica + 1.

`-S or --sort-index`
: Ordene o bloco da árvore índice do mais alto para o mais baixo. Isto otimizará as buscas e tornará a pesquisa em tabela através da chave mais rápida.

`-R or --sort-records=#`
: Ordena o registro de acordo com um índice. Isto faz com que seus dados estejam muito mais localizados e pode aumentar a velocidade das operações `SELECT` e `ORDER BY` neste índice. (Pode ser bem lento na primeira ordenação!) Para encontrar um número de índices da tabela, use `SHOW INDEX`, que exibe os índices de um tabela na mesma ordem que o `myisamchk` os vê. Índices são números que se iniciam com 1.

4.5.6.6 Uso de Memória do `myisamchk`

Alocação de memória é importante quando você executa o `myisamchk`. `myisamchk` não utiliza mais memória do que você especifica com a opção `-O`. Se você irá utilizar o `myisamchk` em grandes arquivos, você deve decidir primeiro quanta memória deseja usar. O valor padrão é utilizar somente 3MB para correções. Utilizando valores maiores, o `myisamchk` pode operar mais rapidamente. Por exemplo, se você tiver mais que 32M de memória RAM, você pode utilizar opções tais como esta (em adição às várias outras que podem ser especificadas):

```
shell> myisamchk -O sort=16M -O key=16M -O read=1M -O write=1M ...
```

Utilizando `-O sort=16M` provavelmente é suficiente para a maioria dos casos.

Certiffique-se que o `myisamchk` utiliza arquivos temporários em `TMPDIR`. Se `TMPDIR` aponta para um sistema de arquivos em memória, você pode facilmente obter erros de memória. Se isto acontecer, configure `TMPDIR` para apontar para algum diretório com mais espaço e reinicie o `myisamchk`.

Quando reparando, o `myisamchk` também precisará de bastante espaço em disco:

- Dobra-se o tamanho do arquivo de registros (o original e uma cópia). Este espaço não é necessário se for feito um reparo com `--quick`, já que neste caso somente o arquivo de índices será recriado. Este espaço é necessário no mesmo disco que se encontra o arquivo de registros original!
- Espaço para o novo arquivo de índice que substitui o antigo. O arquivo de índices antigo é truncando no início, portanto, normalmente este espaço é ignorado. Este espaço é necessário no mesmo disco que o arquivo de índice original!
- Quando utilizando `--recover` ou `--sort-recover` (mas não quando usando `--safe-recover`, será necessário espaço para um buffer de ordenação de: `(maior_chave + tamanho_do_ponteiro_de_registro)*número_de_registros * 2`. Você pode conferir o tamanho das chaves e o tamanho_do_ponteiro_de_registro com `myisamchk -dv tabela`. Este espaço é alocado no disco temporário (especificado por `TMPDIR` ou `--tmpdir=#`).

Se você tiver um problema com espaço em disco durante o reparo, pode-se tentar usar `--safe-recover` em vez de `--recover`.

4.5.6.7 Uso do `myisamchk` para Recuperação em Caso de Falhas

Se você executa o `mysqld` com a opção `--skip-external-locking` (que é o padrão em alguns sistemas, como o Linux), você não pode utilizar com segurança o `myisamchk` para conferir uma tabela se o `mysqld` estiver utilizando a mesma tabela. Se você pode ter certeza que ninguém está acessando as tabelas através do `mysqld` enquanto você executa o `myisamchk`, você só tem que executar o `mysqladmin flush-tables` antes de iniciar a verificação das tabelas. Se você não tem certeza, então você deve desligar o `mysqld` enquanto verifica as tabelas. Se você executa o `myisamchk` enquanto o `mysqld` estiver atualizando as tabelas, você pode obter um altera que a tabela está corrompida mesmo se não estiver.

Se você não estiver utilizando `--skip-external-locking`, pode usar o `myisamchk` para conferir as tabelas a qualquer hora. Enquanto você faz isto, todos os clientes que tentarem atualizar a tabela irão esperar até que o `myisamchk` esteja pronto, antes de continuar.

Se você utilizar o `myisamchk` para reparar ou otimizar tabelas, você **DEVE** sempre assegurar que o servidor `mysqld` não esteja utilizando a tabela (Isto também aplica se você utiliza `--skip-external-locking`). Se você não desligar o `mysql`, você deve, pelo menos, fazer um `mysqladmin flush-tables` antes de executar o `myisamchk`. Suas tabelas **podem estar corrompidos** se o servidor e o `myisamchk` acessarem a tabela simultaneamente.

Este capítulo descreve como checar e lidar com dados corrompidos nos bancos de dados MySQL. Se suas tabelas corromperem com frequência deve ser encontrada a razão para isto! Veja Seção A.4.1 [Falhas], Página 925.

A seção de tabelas `MyISAM` contêm motivos do porque uma tabela pode estar corrompida. Veja Seção 7.1.3 [MyISAM table problems], Página 636.

Quando se realizar recuperação devido a falhas, é importante entender que cada tabela `nome_tabela` em um banco de dados corresponde a tres arquivos no diretório do banco de dados:

Arquivo	**Propósito**
'`nome_tabela.frm`'	Arquivo com definições da tabela (form)
'`nome_tabela.MYD`'	Arquivo de dados
'`nome_tabela.MYI`'	Arquivo de índices

Cada um destes três tipos de arquivos está sujeito a corrupção de várias formas, mas problemas ocorrem mais frequentemente em arquivos de dados e índices.

O `myisamchk` trabalha criando uma cópia do arquivo de dados '`.MYD`' linha a linha. Ele termina o estágio de reparos removendo o antigo arquivo '`.MYD`' e renomeando o novo arquivo com nome original. Se for utilizada a opção `--quick`, `myisamchk` não cria um arquivo '`.MYD`' temporário, mas assume que o arquivo '`.MYD`' está correto e somente gera um novo arquivo índice sem mexer no arquivo de dados. Isto é seguro, pois o `myisamchk` detecta automaticamente se o arquivo '`.MYD`' está corrompido e aborda o reparo neste caso. Você pode também fornecer duas opções `--quick` para o `myisamchk`. Neste caso, o `myisamchk` não aborta em alguns erros (como chaves duplicadas) mas tenta resolvê-los modificando o arquivo '`.MYD`'. Normalmente o uso de duas opções `--quick` é útil somente se você tiver muito pouco espaço em disco para realizer um reparo normal. Neste caso você deve pelo menos fazer um backup antes de executar o `myisamchk`.

4.5.6.8 Como Verificar Erros em Tabelas

Para conferir uma tabela MyISAM, utilize os seguintes comandos:

`myisamchk nome_tabela`
: Encontra 99.99% de todos os erros. O que ele não pode encontrar é corrompimento que envolva **SOMENTE** o arquivo de dados (que não é comum). Se você desejar conferir uma tabela, você deve executar normalmente o `myisamchk` sem opções ou com as opções `-s` ou `--silent`.

`myisamchk -m nome_tabela`
: Encontra 99.999% de todos os erros. Ele verifica primeiramente erros em todas as entradas do índice e então le todos os registros. Ele calcula um checksum para todas as chaves nos registros e verifica se o checksum é o mesmo que o checksum das chaves na árvore de índices.

`myisamchk -e nome_tabela`
: Realiza a verificação completa de todos os dados (`-e` significa "conferência extendida"). Ele faz uma conferência lendo todas as chaves de cada registro para verificar se eles realmente apontam para o registro correto. Isto pode demorar MUITO tempo em uma tabela grande com várias chaves. `myisamchk` normalmente irá parar depois do primeiro erro que encontrar. Se você deseja obter mais informações, pode adicionar a opção `--verbose` (`-v`). Isto faz o `myisamchk` continuar a percorrer a tabela até um máximo de 20 erros. Em utilização normal, um simples `myisamchk` (sem argumentos além do nome da tabela) é suficiente.

`myisamchk -e -i nome_tabela`
: Como o comando anterior, mas a opção `-i` diz ao `myisamchk` para exibir algumas informações estatísticas também.

4.5.6.9 Como Reparar Tabelas

Na seção seguinte nós só falaremos do uso do `myiasmchk` em tabelas `MyISAM` (extensões `.MYI` e `.MYD`). Se você estiver usando tabelas `ISAM` (extensões `.ISM` e `.ISD`), você deve usar a ferramenta `isamchk`.

A partir do MySQL versão 3.23.14, você pode reparar tabelas MyISAM com o comando `REPAIR TABLE`. Veja Seção 4.5.5 [REPAIR TABLE], Página 279.

Os sintomas de uma tabela corrompida incluem pesquisas que abortam inesperadamente e erros como estes:

- '`nome_tabela.frm`' is locked against change
- Can't find file '`nome_tabela.MYI`' (Errcode: ###)
- Unexpected end of file
- Record file is crashed
- Got error ### from table handler

  Para obter mais informações sobre o erro você pode executar `perror ###`. Aqui estão os erros mais comuns que indicam um problema com a tabela:

```
shell> perror 126 127 132 134 135 136 141 144 145
126 = Index file is crashed / Wrong file format
127 = Record-file is crashed
132 = Old database file
134 = Record was already deleted (or record file crashed)
135 = No more room in record file
136 = No more room in index file
141 = Duplicate unique key or constraint on write or update
144 = Table is crashed and last repair failed
145 = Table was marked as crashed and should be repaired
```

  Note que o erro 135 (não mais no arquivo de registro), não é um erro que pode ser corrigido por um simples reparo. Neste caso você deve fazer:

```
ALTER TABLE tabela MAX_ROWS=xxx AVG_ROW_LENGTH=yyy;
```

  Você também pode usar esta técnica para o erro 136 (não mais no arquivo de índice).

Em outros casos, você deve reparar suas tabelas. `myisamchk` pode normalmente detectar a maioria dos problemas que ocorrem.

O processo de reparo involve até quatro estágios, descritos abaixo. Antes de começar, você deve mudar para o diretório do banco de dados e conferir as permissões dos arquivos de tabelas. Tenha certeza que eles possam ser lidos pelo usuário do Unix com o qual `mysqld` é executado (e para você, porque você precisa acessar os arquivos que está conferindo). Se não estiverem, você precisa alterar os arquivos, eles também devem ter a permissão de escrita para você.

Se você estiver utilizando o MySQL versão 3.23.16 e superior, você pode (e deve) usar os comandos `CHECK` e `REPAIR` para conferir e corrigir tabelas `MyISAM`. Veja Seção 4.5.4 [CHECK TABLE], Página 278. Veja Seção 4.5.5 [REPAIR TABLE], Página 279.

A seção do manual sobre manutenção de tabelas inclui as opções para `isamchk/myisamchk`. Veja Seção 4.5.6 [Table maintenance], Página 280.

A seguinte seção são para os casos onde o comando acima falhar ou se você desejar usar os recursos extendidos que o `isamchk` e `myisamchk` fornecem.

Se você for reparar uma tabela da linha de comandos, deve primeiro desligar o servidor `mysqld`. Perceba que quando você executa `mysqladmin shutdown` em um servidor remoto, o servidor `mysqld` irá continuar funcionando por um tempo depois do `mysqladmin` retornar, até que todas as queries parem e todas as chaves sejam descarregadas no disco.

Estágio 1: Verificando suas tabelas

Execute `myisamchk *.MYI` ou `myisamchk -e *.MYI` se você tiver tempo disponível. Utilize a opção `-s` (silencioso) para suprimir informações desnecessárias.

Se o servidor `mysqld` parar, deve ser utilizada a opção –update para dizer ao `myisamchk` marcar a tabela como 'checada'.

Você deve reparar somente as tabelas em que o `myisamchk` indicar um erro. Para tais tabelas, vá para o estágio 2.

Se você obter erros estranhos na verficação (como nos erros `out of memory`), ou se o `myisamchk` quebrar, vá para o estágio 3.

Estágio 2: Reparo simples e seguro

NOTA: Se você deseja que os reparos sejam mais rápidos, devem ser usadas as opções: `-O sorf_buffer=# -O key_buffer=#` (onde # seria 1/4 da memória disponível) para todos comandos `isamchk/myisamchk`.

Primeiro, tente usar `myisamchk -r -q nome_tabela` (`-r -q` significa "modo de recuperação rápida"). Ele tentará reparar o arquivo de índice sem mexer no arquivo de dados. Se o arquivo de dados estiver normal e os links apagados apontam nas localizações corretas dentro do arquivo de dados, isto deve funcionar e a tabela será corrigida. Inicie o reparo da próxima tabela. Outra maneira seria utilizar os seguintes procedimentos:

1. Faça um backup do arquivo de dados antes de continuar.
2. Utilize `myisamchk -r nome_tabela` (`-r` significa modo de "recuperação"). Isto removerá registros incorretos e deletados do arquivo de dados e reconstroi o arquivo de índices.
3. Se o passo anterior falhar, utilize `myisamchk --safe-recover nome_tabela`. O modo de recuperação segura utiliza um metódo de recuperação antiga que trata de alguns casos que o modo de recuperação comum não consegue (porém é mais lento).

Se você obter erros estranhos no reparo (como em erros `out of memory`), ou se o `myisamchk` falhar, vá para o estágio 3.

Estágio 3: Reparo difícil

Você só deve atingir este estágio se o primeiro bloco de 16K do arquivo de índice estiver destruído ou conter informações incorretas, ou se o arquivo de índice não existir. Neste caso, é necessário criar um novo arquivo de índice. Faça como a seguir:

1. Mova o arquivo de dados para algum lugar seguro.

2. Use o arquivo de descrição de tabelas para criar novos arquivos (vazios) de dados e índices:

```
shell> mysql nome_bd
mysql> SET AUTOCOMMIT=1;
mysql> TRUNCATE TABLE nome_tabela;
mysql> quit
```

   Se sua versão do MySQL não possuir `TRUNCATE TABLE`, utilize `DELETE FROM nome_tabela`.
3. Copie o antigo arquivo de dados de volta para o novo arquivo de dados criado. (Não só mova o antigo arquivo de volta para o novo arquivo; você deve uma cópia no caso de algo der errado.)

Volte ao estágio 2. `myisamchk -r -q` deve funcionar agora. (Isto não deve ser um loop eterno.)

No `MySQL` 4.0.2 você também pode utilizar `REPAIR ... USE_FRM` o qual realiza todo o procedimento automaticamente.

Estágio 4: Reparo muito difícil

Você deve atingir este estágio somente se o arquivo de descrição também falhar. Isto nunca deve acontecer, porque o arquivo de descrição não é alterado depois da tabela ser criada:

1. Restaure o arquivo de descrição de um backup e volte ao estágio 3. Você pode também restaurar o arquivo de índice e voltar ao estágio 2. No último caso, você deve iniciar com `myisamchk -r`.
2. Se você não tem um backup mas sabe exatamente como a tabela foi criada, crie uma cópia da tabela em outro banco de dados. Remova o novo arquivo de dados, e então mova a descrição e arquivos de índice do outro banco de dados para o banco de dados com problemas. Isto lhe fornece um novo arquivos índice e descrição, mas mantêm o arquivo de dados da mesma forma. Volte ao estágio 2 e tente reconstruir o arquivo de índices.

4.5.6.10 Otimização de Tabelas

Para agrupar registros fragmentados e eliminar perda de espaço resultante de remoções ou atualizações de registros, execute `myisamchk` no modo de recuperação:

```
shell> myisamchk -r nome_tabela
```

Você pode otimizar uma tabela da mesma forma utilizando a instrução SQL `OPTIMIZE TABLE`. `OPTIMIZE TABLE` faz o reparo de tabelas, analisa chaves e também ordena a árvore de índices para fazer pesquisas por chave mais rápidas. Também não existem possibilidade de interação não desejável entre o utilitário e o servidor, porque o servidor faz todo o trabalho quando você utiliza `OPTIMIZE TABLE`. Veja Seção 4.6.1 [OPTIMIZE TABLE], Página 298.

`myisamchk` também tem um número de outras opção que podem ser usadas para melhorar a performance de uma tabela:

- `-S`, `--sort-index`
- `-R index_num`, `--sort-records=index_num`
- `-a`, `--analyze`

Para uma descrição completa da opção. Veja Seção 4.5.6.1 [myisamchk syntax], Página 281.

4.5.7 Configurando um Regime de Manutenção das Tabelas

A partir do MySQL Versão 3.23.13, você pode conferir tabelas MyISAM com o comando `CHECK TABLE`. Veja Seção 4.5.4 [CHECK TABLE], Página 278. Você pode reparar tabelas com o comando `REPAIR TABLE`. Veja Seção 4.5.5 [REPAIR TABLE], Página 279.

É uma boa idéia verificar as tabelas regularmente em vez de esperar que ocorram problemas. Para propósitos de manutenção você pode utilizar o `myisamchk -s` para verificar as tabelas. A opção `-s` (abreviação de `--silent`) faz com que o `myisamchk` execute em modo silencioso, exibindo mensagens somente quando ocorrem erros.

É também uma boa idéia verificar as tabelas quando o servidor inicia. Por exemplo, sempre que a máquina reinicia no meio de uma atualização, você normalmente precisará conferir todas as tabelas que podem ter sido afetadas. (Isto é uma"tabela com falhas esperadas".) Você pode adicionar um teste ao `mysqld_safe` que executa `myisamchk` para conferir todas tabelas que foram modificadas durante as últimas 24 horas se existir um arquivo '`.pid`' (process ID) antigo depois do último reboot. (O arquivo '`.pid`' é criado pelo `mysqld` quando ele inicia e removido quando ele termina normalmente. A presença de um arquivo '`.pid`' durante a inicialização do sistema indica que o `mysqld` terminou de forma anormal.)

Um teste ainda melhor seria verificar qualquer tabela cuja a data da última modificação é mais recente que a do arquivo '`.pid`'.

Você também deve verificar suas tabelas regularmente durante a operação normal do sistema. Na MySQL AB, nós executamos uma tarefa agendada `cron` para conferir todas nossas tabelas importantes uma vez por semana utilizando uma linha com esta no arquivo '`crontab`':

```
35 0 * * 0 /diretório/do/myisamchk --fast --silent /diretório/de/dados/*/*.MYI
```

Isto exibe informações sobre tabelas com falhas para que possamos examiná-las e repará-las quando necessário.

Como nós não estamos tendo tabelas com falhas inesperadas (tabelas corrompidas por razões diferentes de problemas de hardware) por vários anos (isto realmente é verdade), uma vez por semana é mais que suficiente para nós.

Nós recomendamos que para iniciar, você execute `myisamchk -s` a cada noite em todas as tabelas que foram atualizadas durantes as últimas 24 horas, até que você confie no MySQL como nós confiamos.

Normalmente você não precisará de tanta manutenção em suas tabelas MySQL. Se você estiver alterando tabelas com registros de tamanho dinâmico (tabelas com colunas `VARCHAR`, `BLOB` ou `TEXT`) ou tem tabelas com vários registros apagados você pode desejar de tempos em tempos (uma vez ao mês?) desfragmentar/recuperar espaço das tabelas.

Você pode fazer isto utilizando `OPTIMIZE TABLE` nas tabelas em questão ou se você puder desligar o servidor `mysqld` por um tempo faça:

```
isamchk -r --silent --sort-index -O sort_buffer_size=16M */*.ISM
myisamchk -r --silent --sort-index  -O sort_buffer_size=16M */*.MYI
```

4.5.8 Obtendo Informações sobre as Tabelas

Para obter uma descrição de uma tabela ou estatísticas sobre ela, utilize os comandos mostrados abaixo, nós explicaremos algumas das informações em mais detalhes posteriormente:

- myisamchk -d nome_tabela Executa o `myisamchk` no "modo descritivo" para produzir uma descrição de sua tabela. Se você iniciar o servidor MySQL utilizando a opção `--skip-locking`, `myisamchk` pode relatar um erro para uma tabela que está sendo atualizada enquanto é executado. Entretanto, como o `myisamchk` não altera a tabela no modo de descrição, não existem riscos de destruição de dados.
- myisamchk -d -v nome_tabela Para produzir mais informações sobre o que `myisamchk` está fazendo, adicione `-v` para solicitar a execução em modo verbose.
- myisamchk -eis nome_tabela Exibe somente as informações mais importantes de uma tabela. Ele é lento porque é necessário ler a tabela inteira.
- myisamchk -eiv nome_tabela Isto se parece com `-eis`, mas lhe diz o que está sendo feito.

Exemplo da saída de `myisamchk -d`

```
MyISAM file:     company.MYI
Record format:   Fixed length
Data records:    1403698  Deleted blocks:         0
Recordlength:    226

table description:
Key Start Len Index   Type
1   2     8   unique  double
2   15    10  multip. text packed stripped
3   219   8   multip. double
4   63    10  multip. text packed stripped
5   167   2   multip. unsigned short
6   177   4   multip. unsigned long
7   155   4   multip. text
8   138   4   multip. unsigned long
9   177   4   multip. unsigned long
    193   1           text
```

Exemplo da saída de `myisamchk -d -v` :

```
MyISAM file:         company
Record format:       Fixed length
File-version:        1
Creation time:       1999-10-30 12:12:51
Recover time:        1999-10-31 19:13:01
Status:              checked
Data records:           1403698  Deleted blocks:              0
Datafile parts:         1403698  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     3
Max datafile length: 3791650815  Max keyfile length: 4294967294
```

```
Recordlength:               226

table description:
Key Start Len Index   Type                     Rec/key     Root Blocksize
1   2     8   unique  double                         1 15845376      1024
2   15    10  multip. text packed stripped           2 25062400      1024
3   219   8   multip. double                        73 40907776      1024
4   63    10  multip. text packed stripped           5 48097280      1024
5   167   2   multip. unsigned short              4840 55200768      1024
6   177   4   multip. unsigned long               1346 65145856      1024
7   155   4   multip. text                        4995 75090944      1024
8   138   4   multip. unsigned long                 87 85036032      1024
9   177   4   multip. unsigned long                178 96481280      1024
    193   1           text
```

Exemplo da saída de `myisamchk -eis`:

```
Checking MyISAM file: company
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:  98%  Packed:   17%

Records:          1403698    M.recordlength:     226
Packed:             0%
Recordspace used:     100%   Empty space:          0%
Blocks/Record:   1.00
Record blocks:    1403698    Delete blocks:        0
Recorddata:     317235748    Deleted data:         0
Lost space:             0    Linkdata:             0

User time 1626.51, System time 232.36
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 627, Swaps 0
Blocks in 0 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 639, Involuntary context switches 28966
```

Exemplo da saída de `myisamchk -eiv`:

```
Checking MyISAM file: company
Data records: 1403698   Deleted blocks:       0
- check file-size
- check delete-chain
block_size 1024:
```

```
index  1:
index  2:
index  3:
index  4:
index  5:
index  6:
index  7:
index  8:
index  9:
No recordlinks
- check index reference
- check data record references index: 1
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 2
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
- check data record references index: 3
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
- check data record references index: 5
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 6
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 7
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 8
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 9
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:   9%  Packed:   17%

- check records and index references
[LOTS OF ROW NUMBERS DELETED]

Records:          1403698    M.recordlength:     226   Packed:             0%
Recordspace used:     100%   Empty space:          0%  Blocks/Record:   1.00
Record blocks:    1403698    Delete blocks:        0
Recorddata:     317235748    Deleted data:         0
Lost space:             0    Linkdata:             0

User time 1639.63, System time 251.61
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 10580, Swaps 0
Blocks in 4 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 10604, Involuntary context switches 122798
```

Aqui estão os tamanhos dos arquivos de dados e índices para a tabela utilizada nos exemplos anteriores:

```
-rw-rw-r--   1 monty    tcx     317235748 Jan 12 17:30 company.MYD
-rw-rw-r--   1 davida   tcx      96482304 Jan 12 18:35 company.MYM
```

Explicações para os tipos de informações que o `myisamchk` produz são fornecidas abaixo. O "keyfile" é o arquivo de índices. "Registro" e "linha" são sinônimos:

- ISAM file Nome do arquivo (índice) ISAM.
- Isam-version Versão do formato ISAM. Atualmente sempre 2.
- Creation time Quando o arquivo de dados foi criado.
- Recover time Quando foi a última vez que o arquivo de índices/dados foi reconstruído.
- Data records Quantos registros existem na tabela.
- Deleted blocks Quantos blocos apagados continuam alocando espaço. Você pode otimizar sua tabela para minimizar este espaço. Veja Seção 4.5.6.10 [Otimização], Página 291.
- Datafile: Parts Para formato de registros dinâmicos, isto indica quantos blocos de dados existem. Para uma tabela otimizada sem registros fragmentados, isto é o mesmo que `Data records`.
- Deleted data Quantos bytes de dados deletados não recuperados existem. Você pode otimizar sua tabela para minimizar este espaço. Veja Seção 4.5.6.10 [Otimização], Página 291.
- Data file pointer O tamanho do ponteiro do arquivo de dados, em bytes. Ele normalmente possui 2, 3, 4 ou 5 bytes. A maioria das tabelas trabalham com 2 bytes, mas isto ainda não pode ser controlado pelo MySQL ainda. Para tabelas fixas, isto é um endereço de registro. Para tabelas dinâmicas, isto é um endereço de byte.
- Keyfile pointer O tamanho de um ponteiro de arquivo de índices, em bytes. Ele normalmente possui 1, 2 ou 3 bytes. A maioria das tabelas trabalham com 2 bytes, mas isto é calculado automaticamente pelo MySQL. Ele é sempre um endereço de bloco.
- Max datafile length Qual tamanho o arquivo de dados (arquivos `.MYD`) pode atingir, em bytes.
- Max keyfile length Qual tamanho o arquivo de índices (`.MYI` pode atingir, em bytes.
- Recordlength Quanto espaço cada registro ocupa, em bytes.
- Record format O formato utilizado para armazenar as linhas da tabelas. Os exemplos anteriores abaixo utilizam `Fixed length` (tamanho fixo). Outros valores possíveis são `Compressed`(compactado) e `Packed`(empacotado).
- table description Uma lista de todas as chaves na tabela. Para cada chave, alguma informação de baixo nível é apresentada:

  `Key` O Número desta chave.

  `Start` Onde, no registro, esta parte do índice inicia.

  `Len` Qual o tamanho desta parte do índice. Para números empacotados, isto deve sempre ser o tamanho total da coluna. Para strings, deve ser mais curto que o tamanho total da coluna indexada, porque você pode indexar um prefixo de uma coluna string.

`Index` `unique` ou `multip.` (multiplos). Indica se um valor pode ou não exisitir várias vezes neste índice.

`Type` Que tipo de dados esta parte do índice tem. Isto é um tipo de dados ISAM com as opções `packed`, `stripped` ou `empty`.

`Root` Endereço do bloco de índice raiz.

`Blocksize`
O tamanho de cada bloco de índice. O tamanho padrão é 1024, mas o valor pode ser alterado na compilação.

`Rec/key` Este é um valor estatístico utilizado pelo otimizador. Ele diz quantos registros existem por valor para esta chave. Uma chave única sempre tem um valor de 1. Ele pode ser atualizado depois que uma tabela é carregada (ou muito alterada) com `myisamchk -a`. Se isto não for completamente atualizado, um valor padrão de 30 é fornecido.

- No primeiro exemplo acima, a nona chave é uma chave multi partes com duas partes.
- Keyblocks used Qual o percentual de bloco de chaves são usados. Como a tabela usada nos exemplos foi reorganizada com `myisamchk`, os valores são muito altos (muito próximos do máximo teórico).
- Packed O MySQL tenta empacotar chaves com um sufixo comum. Isto pode ser usado somente para chaves `CHAR/VARCHAR/DECIMAL`. Para strings grandes como nomes, isto pode reduzir significativamente o espaço utilizado. No terceiro exemplo acima, a quarta chave possui 10 caracteres e uma redução de 60% no espaço é obtida.
- Max levels Qual a profundidade da árvore-B para esta chave. Grandes tabelas com chaves longas resultam em valores altos.
- Records Quantos registros existem na tabela.
- M.recordlength A média de tamanho do registro. Para tabelas com registros de tamanho fixo, isto é o tamanho exato do registro.
- Packed O MySQL corta espaços do final de strings. O valor `Packed` indica o percentual de economia alcançado fazendo isto.
- Recordspace used Qual percentual do arquivo de dados é usado.
- Empty space Qual percetual do arquivo de dados não é usado.
- Blocks/Record Número médio de blocos por registro (isto é, de quantos links um registro fragmentado é composto). Sempre será 1 para tabelas de formato fixo. Este valor deve permanecer o mais próximo possível de 1.0. Se ele aumentar, você pode reorganizar a tabela com `myisamchk`. Veja Seção 4.5.6.10 [Otimização], Página 291.
- Recordblocks Quantos blocos (links) são utilizados. Para formatos fixos, este é o mesmo que o número de registros.
- Deleteblocks Quantos blocos (links) foram excluídos.
- Recorddata Quantos bytes no arquivo de dados são usados.
- Deleted data Quantos bytes no arquivo de dados foram apagados (sem uso).
- Lost space Se um registro é atualizado para um tamanho menor, algum espaço é perdido. Isto é a soma de todas estas perdas, em bytes.

- Linkdata Quando o formato de tabela dinâmica é utilizado, fragmentos de registros são ligados com ponteiros (4 a 7 bytes cada). `Linkdata` é a soma do montante de armazenamento utilizado por todos estes ponteiros.

Se uma tabela foi compactada com `myisampack`, `mysiamchk -d` exibe informações adicionais sobre cada coluna da tabela. Veja Seção 4.8.4 [`myisampack`], Página 336, para um exemplo desta informação e uma descrição do que ela significa.

4.6 Adiministração do Banco de Dados e Referência de Linguagem

4.6.1 Sintaxe de `OPTIMIZE TABLE`

```
OPTIMIZE [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name[,tbl_name]...
```

`OPTIMIZE TABLE` deve ser usado se você apagou uma grande parte de uma tabela ou se você fez várias alterações à uma tabela com registros de tamanho variável (tabelas que tenham campos do tipo `VARCHAR`, `BLOB` ou `TEXT`). Registros apagados são mantidos em uma lista de ligações e operações `INSERT` subsequentes reutilizam posições de registros antigos. Você pode utilizar `OPTIMIZE TABLE` para reclamar o espaço inutilizado e para desfragmentar o arquivo de dados.

Na maioria da configurações você não tem que executar `OPTIMIZE TABLE`. Mesmo se você fizer diversas atualizações para registros de tamanhos variáveis não é desejável que você precise fazer isto mais que uma vez por mês/semana e apenas em determinadas tabelas.

No momento `OPTIMIZE TABLE` só funciona em tabelas **MyISAM** e `BDB`. Para tabelas `BDB`, `OPTIMIZE TABLE` é atualmente mapeado para `ANALIZE TABLE`. Veja Seção 4.6.2 [ANALYZE TABLE], Página 298.

Você pode ter a otimização de tabelas trabalhando em outros tipos de tabelas iniciando o `mysqld` com `--skip-new` ou `--safe-mode`, mas neste caso, `OPTIMIZE TABLE` é mapeado apenas para `ALTER TABLE`.

`OPTIMIZE TABLE` funciona da seguinte forma:

- Se a tabela tem registros excluídos ou divididos, repara a tabela.
- Se as páginas de índice não estão ordenas, ordene-as.
- Se as estatísticas não estão atualizadas (e o reparo não pode ser feito ordenando o índice), atualize-as.

Perceba que a tabela estará bloqueada durante o tempo em que `OPTIMIZE TABLE` estiver executando.

Antes do MySQL 4.1.1, o `OPTIMIZE` comnado não gravava no log binário. Desde o MySQL 4.1.1 eles são escritos no log binário a menos que a palavra chave opcional `NO_WRITE_TO_BINLOG` (ou se alias `LOCAL`) seja usada.

4.6.2 Sintaxe de `ANALYZE TABLE`

```
ANALYZE [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name[,tbl_name...]
```

Analisa e armazena a distribuição de chaves para a tabela. Durante a análise a tabela é bloqueada com uma trava de leitura. Isto funciona em tabelas `MyISAM` e `BDB`.

Isto seria equivalente a executar `myisamchk -a` na tabela.

O MySQL utiliza a distribuição de chaves armazenadas para decidir em que ordem tabelas devem ser unidas quando alguém faz um join em alguma coisa diferente de uma constante.

O comando retorna uma tabela com as seguintes colunas:

Coluna	**Valor**
Table	Nome da Tabela
Op	Sempre `analyze`
Msg_type	Um dos seguintes: `status`, `error`, `info` ou `warning`
Msg_text	A mensagem

Você pode verificar a distribuição de chaves armazenadas com o comando `SHOW INDEX`. Veja Seção 4.6.8.1 [Show database info], Página 303.

Se a tabela não foi alterada deste o último comando `ANALYZE TABLE`, a tabela não será analisada novamente.

Antes do MySQL 4.1.1, o `ANALYZE` comnado não gravava no log binário. Desde o MySQL 4.1.1 eles são escritos no log binário a menos que a palavra chave opcional `NO_WRITE_TO_BINLOG` (ou se alias `LOCAL`) seja usada.

4.6.3 Sintaxe de `CHECKSUM TABLE`

```
CHECKSUM TABLE tbl_name[,tbl_name ...] [ QUICK | EXTENDED ]
```

Reports a table checksum. If `QUICK` is specified, live table checksum is reported, or `NULL` if the table does not support live checksum. This is very fast. In `EXTENDED` mode the whole table is read row by row and the checksum is calculated. This can be very slow for large tables. By default - with neither `QUICK` nor `EXTENDED` - MySQL returns live checksum if the table support it and scans the table otherwise.

Este comando está implementado no MySQL 4.1.1.

4.6.4 Sintaxe de `FLUSH`

```
FLUSH [LOCAL | NO_WRITE_TO_BINLOG] flush_option [,flush_option] ...
```

Você deve utilizar o comando `FLUSH` se desejar limpar algum dos caches internos que o MySQL usa. Para executar `FLUSH`, você deve ter o privilégio **RELOAD**.

`opções` podem ser qualquer uma das seguintes:

Option	**Description**

`HOSTS`	Esvazia as tabelas de cache de nomes de máquinas. Você deve descarregar as tabelas de nomes de máquinas se alguma de suas máquinas receber um número IP diferente ou se você obter a mensagem de erro `Host ... is blocked`. Quando mais de `max_connect_erros` erros occorrer em um registro para uma determinada máquina enquanto se conecta ao servidor MySQL, o MySQL assume que algo está errado e bloqueia futuras requisições desta máquina. A descarga na tabela de nomes de máquinas permite à máquina se conectar novamente. Veja Seção A.2.5 [Máquina bloqueada], Página 916.) Você pode iniciar o `mysqld` com `-O max_connection_errors=999999999` para evitar esta mensagem de erro.
`DES_KEY_FILE`	Recarrega a chave DES do arquivo que foi especificado com a opção `--des-key-file` durante inicialização do servidor.
`LOGS`	Fecha e reabre todos os arquivos de log. Se você tiver especificado o arquivo de logs de atualizações ou um arquivo de log binário sem uma extensão, o número de extensão do arquivo log será sempre incrementado de um em relação ao arquivo anterior. Se você usou uma extensão no nome do arquivo, o MySQL irá fechar e reabrir o arquivo de log de atualizações. Veja Seção 4.10.3 [Log de atualização], Página 373. Isto é a mesma coisa que enviar o sinal `SIGHUP` para o servidor `mysqld`.
`PRIVILEGES`	Recarrega os privilégios das tabelas de permissões no banco de dados `mysql`.
`QUERY CACHE`	Defragmenta a cache de consulta par utilizar melhor a sua memória. Este comando não remove qualquer consulta da cache, ao contrário de `RESET QUERY CACHE`.
`TABLES`	Fecha todas as tabelas abertas e força o fechamento de todas as tabelas em uso
`[TABLE | TABLES] nome_tabela [,nome_tabela...]`	Descarga somente das tabelas fornecidas.
`TABLES WITH READ LOCK`	Fecha todas tabelas abertas e bloqueia todas tabelas para todos os bancos de dados com leitura até que alguém execute `UNLOCK TABLES`. Isto é uma maneira muito conveniente para fazer backups se você possui um sistema de arquivos, como Veritas, que pode fazer uma imagem instantânea (snapshot) de um certo momento.
`STATUS`	Reinicia a maioria das variáveis de status para zero. Isto é algo que deve ser usado somente para depurar uma consulta.
`USER_RESOURCES`	Zera todos os recirsos dos usuários. Isto permitirá que usuários bloqueados façam login novamente. Veja Seção 4.4.7 [Recursos de usuários], Página 265.

Antes do MySQL 4.1.1, o `FLUSH` comnado não gravava no log binário. Desde o MySQL 4.1.1 eles são escritos no log binário a menos que a palavra chave opcional `NO_WRITE_TO_BINLOG` (ou se alias `LOCAL`) seja usada, ou que o comando contenha um dos argumentos: `LOGS`, `MASTER`, `SLAVE`, `TABLES WITH READ LOCK`, pois qualquer um desses argumwentos podem causar problemas se replicados para um slave.

Você pode também acessar cada um dos comandos vistos acima com o utilitário `mysqladmin`, utilizando os comandos `flush-hosts`, `flush-logs`, `reload` ou `flush-tables`.

Também de uma olhada no comando `RESET` usado com a replicação. Veja Seção 4.6.5 [RESET], Página 301.

4.6.5 Sintaxe de `RESET`

```
RESET reset_option [,reset_option] ...
```

O comando `RESET` é usado para limpar coisas. Ele também atua como uma versão mais forte do comando `FLUSH`. Veja Seção 4.6.4 [FLUSH], Página 299.

Para executar `RESET`, você deve ter o privilégio `RELOAD`.

Opção	Descrição
`MASTER`	Deleta todos os logs binários listados no arquivo índice, esvaziando o arquivo de índice do log binário. Anteriormente chamado `FLUSH MASTER`. Veja Seção 4.11.7 [Replicação], Página 400.
`SLAVE`	Faz o slave "esquecer" a sua posição de replicação no log binário do master. Anteriormente chamado `FLUSH SLAVE`. Veja Seção 4.11.8 [Replicação], Página 402.
`QUERY CACHE`	Remove todos os resulatdos de consultas da cache de consultas.

4.6.6 Sintaxe de `PURGE MASTER LOGS`

```
PURGE {MASTER|BINARY} LOGS TO nome_binlog
PURGE {MASTER|BINARY} LOGS BEFORE data
```

Este comando é usado para deletar todos os logs binários estritamente anteriores ao binlog ou data especificada. Veja Seção 4.11.7 [Replication Master SQL], Página 400.

`PURGE BINARY LOGS` está disponível como um sinônimo para `PURGE MASTER LOGS` a partir do MySQL 4.1.1.

4.6.7 Sintaxe de `KILL`

```
KILL thread_id
```

Cada conexão ao `mysqld` executa em uma thread separada. Você pode ver quais threas estão em execução com o comando `SHOW PROCESSLIST` e matar uma thread com o comando `KILL thread_id`.

Se você tiver o privilégio `PROCESS`, você pode ver todas as threads. Se você tiver o privilégio `SUPER`, você pode matar todas as threads. Caso contrário, você pode ver e matar somente suas próprias threads.

Você também pode usar os comandos `mysqladmin processlist` e `mysqladmin kill` para examinar e matar threads.

Nota: Atualmente você não pode utilizar `KILL` com a biblioteca do servidor MySQL embutido, porque o servidor embutido apenas roda dentro das threads da aplicação, ela não cria threads de conexões por si própria.

Quando você utiliza um `KILL`, um sinal (flag) `kill` especifico é configurado para a thread.

Na maioria dos casos pode levar algum tempo para a thread morrer pois o sinal kill só é checado em intervalos específicos.

- Nos loops `SELECT`, `ORDER BY` e `GROUP BY`, o sinal é checado depois de ler um bloco de registros. Se o sinal kill está habilitado a instrução é abortada.
- Na execução de um `ALTER TABLE` o sinal kill é conferido antes de cada bloco de registros ser lido da tabela original. Se o sinal kill foi habilitado, o comando é abortado e a tabela temporária apagada.
- Ao fazer um `UPDATE TABLE` and `DELETE TABLE`, o sinal de kill é conferido depois de que cada bloco é lido e depois de cada atualização ou remoção de registro. Se o sinal kill está habilitado, a instrução é abortada. Note que se você não estiver utilizando transações, as alterações não irão ser desfeitas!
- `GET_LOCK()` irá aborar com `NULL`.
- Uma thread `INSERT DELAYED` irá rapidamente descarregar todos registros que estiverem em memória e morrer.
- Se a thread estiver no manipulador de bloqueio de tabelas (status: `Locked`), o bloqueio de tabela será abortado rapidamente.
- Se a thread estiver esperando por espaço livre em disco numa chamada `write`, a escrita é abortada com uma mensagem de espaço em disco insuficiente.

4.6.8 Sintaxe de `SHOW`

```
   SHOW DATABASES [LIKE wild]
ou SHOW [OPEN] TABLES [FROM nome_bd] [LIKE wild]
ou SHOW [FULL] COLUMNS FROM nome_tbl [FROM nome_bd] [LIKE wild]
ou SHOW INDEX FROM nome_tbl [FROM nome_bd]
ou SHOW TABLE STATUS [FROM nome_bd] [LIKE wild]
ou SHOW STATUS [LIKE wild]
ou SHOW VARIABLES [LIKE wild]
ou SHOW [BDB] LOGS
ou SHOW [FULL] PROCESSLIST
ou SHOW GRANTS FOR user
ou SHOW CREATE TABLE nome_tbl
ou SHOW MASTER STATUS
ou SHOW MASTER LOGS
ou SHOW SLAVE STATUS
ou SHOW WARNINGS [LIMIT row_count]
ou SHOW ERRORS [LIMIT row_count]
ou SHOW TABLE TYPES
```

SHOW fornece informações sobre bancos de dados, tabelas, colunas ou informações do estado do servidor. Se a parte LIKE wild é usada, a string wild pode ser uma string que usa os meta caracteres '%' e '_' do SQL.

4.6.8.1 Recuperando Informações sobre Bancos de Dados, Tabelas, Colunas e Índices

Você pode usar nome_bd.nome_tabela como uma alternativa para a sintaxe nome_tabela FROM nome_bd. Estas duas declarações são equivalentes:

```
mysql> SHOW INDEX FROM minhatabela FROM meudb;
mysql> SHOW INDEX FROM meubd.minhatabela;
```

SHOW DATABASES lista os bancos de dados no servidor MySQL. Você também pode obter esta lista utilizando o comando mysqlshow. Na versão 4.0.2 você verá apenas aqeules banco de dados para os quais você tem algum tipo de privilégio, se você não tiver o privilégio global SHOW DATABASES.

SHOW TABLES lista as tabelas em um banco de dados específico. Esta lista também pode ser obtida utilizando o comando mysqlshow nome_db.

NOTA: Se um usuário não possui nenhum privilégio para uma tabela, a tabela não será mostrada na saída de SHOW TABLES ou mysqlshow nome_db

SHOW OPEN TABLES lista as tabelas que estão abertas no cache de tabelas. Veja Seção 5.4.7 [Cache de Tabelas], Página 451. O campo Comment diz quantas vezes a tabela está em cached e in_use.

SHOW COLUMNS lista as colunas em uma determinada tabela. Se você especificar a opção FULL, também irá obter os privilégios que você possui para cada coluna. Se os tipos de colunas forem diferentes do que você esperava baseando na declaração CREATE TABLE, perceba que o MySQL algumas vezes altera os tipos das colunas. Veja Seção 6.5.3.1 [Mudança de tipos de colunas], Página 607. A partir do MySQL 4.1, a palavra chave FULL também faz com que qualquer comentário por coluna seja mostrado.

A instrução DESCRIBE fornece informação similar à SHOW COLUMNS. Veja Seção 6.6.2 [DESCRIBE], Página 614.

SHOW FIELDS é um sinônimo para SHOW COLUMNS e SHOW KEYS um sinônimo para SHOW INDEX. Você também pode listar as colunas ou índices de uma tabela com mysqlshow nome_db nome_tabela ou mysqlshow -k nome_bd nome_tabela.

SHOW INDEX retorna a informação de índice em um formato que lembra bem a chamada SQLStatistics do ODBC. As seguintes colunas são retornadas:

Coluna	Significado
Table	Nome da tabela.
Non_unique	0 se o índice não puder conter duplicidades, 1 se puder
Key_name	Nome do índice.
Seq_in_index	Número da sequência da coluna no índice, à partir de 1.
Column_name	Nome da coluna.

`Collation`	Como a coluna é ordenada no índice. No MySQL, pode ter valores '`A`' (Ascendente) ou `NULL` (Not sorted).
`Cardinality`	Número de valores únicos no índice. Isto é atualizado executando `isamchk -a`.
`Sub_part`	Número de caracteres indexados se a coluna só é a indexada parcialmente. `NULL` se a chave inteira for indexada.
`Null`	Contém 'YES' se a coluna puder conter `NULL`.
`Index_type`	Método de índice utilizado.
`Comment`	Vários comentários. No momento, ele diz no MySQL < 4.0.2 se o índice é `FULLTEXT` ou não.

Perceba que como o `Cardinality` é contado baseado nas estatísticas armazenadas como inteiros, ele pode não ser exato para tabelas pequenas.

As colunas `Null` e `Index_type` foram adicionadas no MySQL 4.0.2.

4.6.8.2 SHOW TABLE STATUS

```
SHOW TABLE STATUS [FROM nome_bd] [LIKE wild]
```

`SHOW TABLE STATUS` (introduzido na versão 3.23) funciona como o `SHOW STATUS`, mas fornece muitas informações sobre cada tabela. Você também pode obter esta lista utilizando o comando `mysqlshow --status nome_bd`. As seguintes colunas são retornadas:

Coluna	**Significado**
`Name`	Nome da tabela.
`Type`	Tipo da tabela. Veja Cap"ptexi tulo 7 [Table types], Página 630.
`Row_format`	O formato de armazenamento do registro (Fixed (Fixo), Dynamic(dinâmico), ou Compressed (Compactado)).
`Rows`	Número de registros.
`Avg_row_length`	Tamanho médio do registro.
`Data_length`	Tamanho do arquivo de dados.
`Max_data_length`	Tamanho máximo do arquivo de dados. Para formatos de registro fixo, este é o número maimo de registros na tabela. Para formatos de registro dinâmicos, este é o número total de bytes de dados que pode ser armazenados na tabela, dado o tamanho do ponteiro de dados utilizado.
`Index_length`	Tamanho do arquivo de índice.
`Data_free`	Número de bytes alocados mas não utilizados.
`Auto_increment`	Próximo valor do auto incremento.
`Create_time`	Quando a tabela foi criada.
`Update_time`	A última vez que arquivo de dados foi atualizado.
`Collation`	Conjunto de caracter e collation da tabela. (novo no 4.1.1)
`Checksum`	Valor do checksum (se existir). (novo no 4.1.1)
`Check_time`	A última vez que a tabela foi verificada.
`Create_options`	Opções extras usadas com `CREATE TABLE`.

`Comment` O Comentário utilizado quando a tabela é criada (ou alguma informação do porquê do MySQL não poder acessar a informação da tabela).

Tabelas `InnoDB` irão relatar o espaço livre no tablespace no comentário da tabela.

4.6.8.3 SHOW STATUS

`SHOW STATUS` fornece informações de status do servidor (como `mysqladmin extended-status`). A saída é parecida com o que está exibido abaixo, apesar dos números e formatos provavelmente serem diferentes:

```
+--------------------------+------------+
| Variable_name            | Value      |
+--------------------------+------------+
| Aborted_clients          | 0          |
| Aborted_connects         | 0          |
| Bytes_received           | 155372598  |
| Bytes_sent               | 1176560426 |
| Connections              | 30023      |
| Created_tmp_disk_tables  | 0          |
| Created_tmp_tables       | 8340       |
| Created_tmp_files        | 60         |
| Delayed_insert_threads   | 0          |
| Delayed_writes           | 0          |
| Delayed_errors           | 0          |
| Flush_commands           | 1          |
| Handler_delete           | 462604     |
| Handler_read_first       | 105881     |
| Handler_read_key         | 27820558   |
| Handler_read_next        | 390681754  |
| Handler_read_prev        | 6022500    |
| Handler_read_rnd         | 30546748   |
| Handler_read_rnd_next    | 246216530  |
| Handler_update           | 16945404   |
| Handler_write            | 60356676   |
| Key_blocks_used          | 14955      |
| Key_read_requests        | 96854827   |
| Key_reads                | 162040     |
| Key_write_requests       | 7589728    |
| Key_writes               | 3813196    |
| Max_used_connections     | 0          |
| Not_flushed_key_blocks   | 0          |
| Not_flushed_delayed_rows | 0          |
| Open_tables              | 1          |
| Open_files               | 2          |
| Open_streams             | 0          |
| Opened_tables            | 44600      |
```

```
| Questions                 | 2026873    |
| Select_full_join          | 0          |
| Select_full_range_join    | 0          |
| Select_range              | 99646      |
| Select_range_check        | 0          |
| Select_scan               | 30802      |
| Slave_running             | OFF        |
| Slave_open_temp_tables    | 0          |
| Slow_launch_threads       | 0          |
| Slow_queries              | 0          |
| Sort_merge_passes         | 30         |
| Sort_range                | 500        |
| Sort_rows                 | 30296250   |
| Sort_scan                 | 4650       |
| Table_locks_immediate     | 1920382    |
| Table_locks_waited        | 0          |
| Threads_cached            | 0          |
| Threads_created           | 30022      |
| Threads_connected         | 1          |
| Threads_running           | 1          |
| Uptime                    | 80380      |
+---------------------------+------------+
```

As variáveis de estado listadas acima tem o seguinte significado:

Variável	**Signficado**
`Aborted_clients`	Número de conexões abortadas porque o cliente morreu sem fechar a conexão corretamente. Veja Seção A.2.10 [Erros de Comunicação], Página 918.
`Aborted_connects`	Número de tentativas que falharam ao tentar a conexão ao servidor MySQL. Veja Seção A.2.10 [Erros de Comunicação], Página 918.
`Bytes_received`	Número de bytes recebidos por todos os clientes.
`Bytes_sent`	Número de bytes enviados para todos os clientes..
`Com_xxxx`	Número de vezes que os comandos xxx foram executados.
`Connections`	Número de tentativas de conexão ao servidor MySQL.
`Created_tmp_disk_tables`	Número de tabelas temporárias implicitas em disco criadas durante a execução de instruções.
`Created_tmp_tables`	Número de tabelas temporárias implicitas na memória criadas durante execuções de instruções.
`Created_tmp_files`	Quantos arquivos temporários o `mysqld` criou.
`Delayed_insert_threads`	Número de threads para tratamento de insertdelayed que estão em uso.
`Delayed_writes`	Número de registros escritos com `INSERT DELAYED`.
`Delayed_errors`	Número de registros escritos com `INSERT DELAYED` onde algum erro ocorreu (provavelmente `duplicate key`).
`Flush_commands`	Número de comandos `FLUSH` executados.
`Handler_delete`	Número de vezes que um registro foi apagado da tabela.

`Handler_read_first`	Número de vezes que a primeira entrada foi lida de um índice. Se este valor for alto, sugere que o servidor está fazendo várias leituras de índices, por exemplo, `SELECT col1 FROM foo`, assumindo que col1 é indexado.
`Handler_read_key`	Número de requisições para ler um registro baseado em uma chave. Se este valor for alto, é uma boa indicação que suas pesquisas e tabelas estão indexadas corretamente.
`Handler_read_next`	Número de requisições para ler o próximo registro na ordem da chave. Este valor será aumentado se você consultar uma coluna de índice com uma faixa restrita. Ele também aumentará se forem feitas busca nos índices.
`Handler_read_prev`	Némro de requisições ao registros anterior na ordem da chave. Ele é principalmente usado para otimizar `ORDER BY ... DESC`.
`Handler_read_rnd`	Número de requisições para ler um registro baseado em uma posição fixa. O valor será alto se você estiver executando várias pesquisas que exigem ordenação do resultado.
`Handler_read_rnd_next`	Número de requisões para ler o próximo registro no arquivo de dados. Será alto se você estiver fazendo várias buscas na tabela. Geralmente sugere que suas tabelas não estão corretamente indexadas ou que suas pesquisas não foram escritas para tirar vantagem dos índices existentes.
`Handler_rollback`	Números de comandos `ROLLBACK` internos.
`Handler_update`	Número de requisições para atualizar um registro em uma tabela.
`Handler_write`	Número de requisições para inserir um registro em uma tabela.
`Key_blocks_used`	O número de blocos utilizados no cache das chaves.
`Key_read_requests`	O número de requisições para ler um bloco de chaves do cache.
`Key_reads`	O número de leituras físicas de blocos de chaves do disco.
`Key_write_requests`	O número de requisições para gravar um bloco de chaves no cache.
`Key_writes`	O número de escritas físicas de um bloco de chaves para o disco.
`Max_used_connections`	O número máximo de conexões simultâneas que foram usadas.
`Not_flushed_key_blocks`	Blocos de chaves no cache de chaves que foi alterado mas ainda não foi descarregado para o disco.
`Not_flushed_delayed_rows`	Número de registros esperando para serem escritos em filas `INSERT DELAY`.
`Open_tables`	Número de tabelas abertas.
`Open_files`	Número de arquivos abertos.
`Open_streams`	Número de fluxos abertos (usado principalmente para logs).
`Opened_tables`	Número de tabelas que foram abertas.
`Rpl_status`	Status de replicação segura. (Ainda não está em uso).
`Select_full_join`	Número de joins sem chaves (Se for 0, você deve conferir com cuidado o índice de suas tabelas).
`Select_full_range_join`	Número de joins onde foram usadas pesquisas segmentadas na tabela de referencia.
`Select_range`	Número de joins onde foram usadas faixas da primeira tabela. (Normalmente não é crítica mesmo se o valor estiver alto.)

`Select_scan`	Número de joins onde fizemos uma busca completa na primeira tabela.
`Select_range_check`	Número de joins sem chaves onde o uso de chave foi conferido após cada registro (Se for 0, o índice de suas tabelas deve ser conferido com cuidado)
`Questions`	Número de consultas enviadas para o servidor.
`Slave_open_temp_tables`	Número de tabelas temporárias atualmente abertas pela thread slave.
`Slave_running`	É `ON` se este slave está conectado a um master.
`Slow_launch_threads`	Número de threads que levaram mais tempo do que `slow_lauch_time` para serem criadas.
`Slow_queries`	Número de consultas que levaram mais tempo que `long_query_time` segundos. Veja Seção 4.10.5 [Log de consultas lentas], Página 377.
`Sort_merge_passes`	Número de ifusões feitas pelo algorítmo de ordenação. Se este valor for alto você deve considerar o aumento de `sort_buffer`.
`Sort_range`	Número de ordenações que foram feitas com limites.
`Sort_rows`	Número de registros ordenados.
`Sort_scan`	Número de ordenações que foram feitas lendo a tabela.
`ssl_xxx`	Variáveis usadas por SSL; Ainda não implementado.
`Table_locks_immediate`	Número de vezes que um travamento de tabela foi obtido de maneira automática.
`Table_locks_waited`	Número de vezes que um bloqueio de tabela não pôde ser obtido imediatamente e foi preciso esperar. Se o valor for alto, e você tiver problemas de performance, suas consultas devem ser otimizadas e depois dividir sua tabela ou tabelas ou usar replicação. Disponível à partir da versão 3.23.33
`Threads_cached`	Número de threads no cache de threads.
`Threads_connected`	Número de conexões atuais abertas.
`Threads_created`	Número de threads criadas para lidar com conexões.
`Threads_running`	Número de threads que não estão dormindo.
`Uptime`	Quantos segundos o servidor está funcionando.

Alguns comentários sobre a tabela acima:

- Se `Opened_tables` for grande, provavelmente sua variável `table_cache` está muito pequena.
- Se `key_reads` for grande, provavelmente sua variável `key_buffer_size` provavelmente está muito pequena. O índice de acertos do cache pode ser calculaldo com `key_reads/key_read_requests`.
- Se `Handler_read_rnd` for grande, provavelmente você possui várias consultas que exigem do MySQL fazer busca em tabelas inteiras ou você tem joins que não utilizam chaves corretamente.
- Se `Threads_created` for grande você pode desejar aumentar a variável `thread_cache_size`. A taxa de acerto da cache pode ser calculada com `Threads_created/Connections`.
- Se `Created_tmp_disk_tables` for grande, você pode querer aumentar a variável `tmp_table_size` par obter tabelas temporárias em memórias em vez de tabelas em disco.

4.6.8.4 SHOW VARIABLES

```
SHOW [GLOBAL | SESSION] VARIABLES [LIKE wild]
```

`SHOW VARIABLES` exibe os valores de algumas variáveis de sistema do MySQL.

As opções `GLOBAL` e `SESSION` são novas no MySQL 4.0.3. Com `GLOBAL` você obterá as variáveis que serão utilizadas para novas conexões ao MySQL. Com `SESSION` você obterá os valores que estão em efeito para a conexão atual. Se você não estiver usando nenhuma opção, `SESSION` será usada.

Se os valores padrões não lhe servirem, você pode configurar a maioria destas variáveis usando as opções de linha de comando na inicialização do `mysqld`. Veja Seção 4.1.1 [Command-line options], Página 207. Você pode alterar a maioria das variáveis com o comando `SET`. Veja Seção 5.5.6 [SET], Página 460.

A saída de `SHOW VARIABLES` se parece com o exibido abaixo, embora o formato e os números possam divergir. Você também pode conseguir esta informação usando o comando `mysqladmin variables`.

```
+---------------------------------+-----------------------------+
| Variable_name                   | Value                       |
+---------------------------------+-----------------------------|
| back_log                        | 50                          |
| basedir                         | /usr/local/mysql            |
| bdb_cache_size                  | 8388572                     |
| bdb_log_buffer_size             | 32768                       |
| bdb_home                        | /usr/local/mysql            |
| bdb_max_lock                    | 10000                       |
| bdb_logdir                      |                             |
| bdb_shared_data                 | OFF                         |
| bdb_tmpdir                      | /tmp/                       |
| bdb_version                     | Sleepycat Software: ...   |
| binlog_cache_size               | 32768                       |
| bulk_insert_buffer_size         | 8388608                     |
| character_set                   | latin1                      |
| character_sets                  | latin1 big5 czech euc_kr  |
| concurrent_insert               | ON                          |
| connect_timeout                 | 5                           |
| convert_character_set           |                             |
| datadir                         | /usr/local/mysql/data/      |
| delay_key_write                 | ON                          |
| delayed_insert_limit            | 100                         |
| delayed_insert_timeout          | 300                         |
| delayed_queue_size              | 1000                        |
| flush                           | OFF                         |
| flush_time                      | 0                           |
| ft_boolean_syntax               | + -><()~*:""&|              |
| ft_min_word_len                 | 4                           |
| ft_max_word_len                 | 84                          |
| ft_query_expansion_limit        | 20                          |
```

```
| ft_stopword_file                | (built-in)                  |
| have_bdb                        | YES                         |
| have_innodb                     | YES                         |
| have_isam                       | YES                         |
| have_raid                       | NO                          |
| have_symlink                    | DISABLED                    |
| have_openssl                    | YES                         |
| have_query_cache                | YES                         |
| init_file                       |                             |
| innodb_additional_mem_pool_size | 1048576                     |
| innodb_buffer_pool_size         | 8388608                     |
| innodb_data_file_path           | ibdata1:10M:autoextend      |
| innodb_data_home_dir            |                             |
| innodb_file_io_threads          | 4                           |
| innodb_force_recovery           | 0                           |
| innodb_thread_concurrency       | 8                           |
| innodb_flush_log_at_trx_commit  | 1                           |
| innodb_fast_shutdown            | ON                          |
| innodb_flush_method             |                             |
| innodb_lock_wait_timeout        | 50                          |
| innodb_log_arch_dir             |                             |
| innodb_log_archive              | OFF                         |
| innodb_log_buffer_size          | 1048576                     |
| innodb_log_file_size            | 5242880                     |
| innodb_log_files_in_group       | 2                           |
| innodb_log_group_home_dir       | ./                          |
| innodb_mirrored_log_groups      | 1                           |
| interactive_timeout             | 28800                       |
| join_buffer_size                | 131072                      |
| key_buffer_size                 | 16773120                    |
| language                        | /usr/local/mysql/share/...  |
| large_files_support             | ON                          |
| local_infile                    | ON                          |
| locked_in_memory                | OFF                         |
| log                             | OFF                         |
| log_update                      | OFF                         |
| log_bin                         | OFF                         |
| log_slave_updates               | OFF                         |
| log_slow_queries                | OFF                         |
| log_warnings                    | OFF                         |
| long_query_time                 | 10                          |
| low_priority_updates            | OFF                         |
| lower_case_table_names          | OFF                         |
| max_allowed_packet              | 1047552                     |
| max_binlog_cache_size           | 4294967295                  |
| max_binlog_size                 | 1073741824                  |
| max_connections                 | 100                         |
```

```
| max_connect_errors              | 10                          |
| max_delayed_threads             | 20                          |
| max_heap_table_size             | 16777216                    |
| max_join_size                   | 4294967295                  |
| max_relay_log_size              | 0                           |
| max_sort_length                 | 1024                        |
| max_user_connections            | 0                           |
| max_tmp_tables                  | 32                          |
| max_write_lock_count            | 4294967295                  |
| myisam_max_extra_sort_file_size | 268435456                   |
| myisam_repair_threads           | 1                           |
| myisam_max_sort_file_size       | 2147483647                  |
| myisam_recover_options          | force                       |
| myisam_sort_buffer_size         | 8388608                     |
| net_buffer_length               | 16384                       |
| net_read_timeout                | 30                          |
| net_retry_count                 | 10                          |
| net_write_timeout               | 60                          |
| open_files_limit                | 1024                        |
| pid_file                        | /usr/local/mysql/name.pid   |
| port                            | 3306                        |
| protocol_version                | 10                          |
| query_cache_limit               | 1048576                     |
| query_cache_size                | 0                           |
| query_cache_type                | ON                          |
| read_buffer_size                | 131072                      |
| read_rnd_buffer_size            | 262144                      |
| rpl_recovery_rank               | 0                           |
| safe_show_database              | OFF                         |
| server_id                       | 0                           |
| slave_net_timeout               | 3600                        |
| skip_external_locking           | ON                          |
| skip_networking                 | OFF                         |
| skip_show_database              | OFF                         |
| slow_launch_time                | 2                           |
| socket                          | /tmp/mysql.sock             |
| sort_buffer_size                | 2097116                     |
| sql_mode                        |                             |
| table_cache                     | 64                          |
| table_type                      | MYISAM                      |
| thread_cache_size               | 3                           |
| thread_stack                    | 131072                      |
| tx_isolation                    | READ-COMMITTED              |
| timezone                        | EEST                        |
| tmp_table_size                  | 33554432                    |
| tmpdir                          | /tmp/:/mnt/hd2/tmp/         |
| version                         | 4.0.4-beta                  |
```

```
| wait_timeout                    | 28800                          |
+---------------------------------+--------------------------------+
```

Cada opção é descrita abaixo. Valores para tamanhos de buffer, comprimento e tamanho de pilha são fornecidos em bytes. Você pode especificar valores com sufixos 'K' ou `M` para indicar o valor em kilobytes ou megabytes. Por exemplo, `16M` indica 16 Megabytes. Não importa se os sufixos estão em letras maiúsuculas ou minúsculas; `16M` e `16m` são equivalentes:

- `ansi_mode`. Está ligado (`ON`) se o `mysqld` foi iniciado com `--ansi`. Veja Seção 1.8.2 [Modo ANSI], Página 42.
- `back_log` O número de requisições de conexões que o MySQL pode suportar. Isto entra em cena quando a thread principal do MySQL recebe **MUITAS** solicitações de conexões em um espaço curto de tempo. Eles tomam algum tempo (porém muito pouco) da a thread principal para conferir a conexão e iniciar uma nova thread. O valor `back_log` indica quantas requisições podem ser empilhadas durante este breve tempo antes do MySQL parar de responder a novas requisições. Você isó precisa aumentá-lo se espera um número alto de conexões em um curto período de tempo

  Em outras palavras, este valor é o tamanho da fila de escuta para novas conexões TCP/IP. Seu sistema operacional tem o próprio limite para o tamanho desta fila. A página do manual Unix da chamada de sistema `listen(2)` deve fornecer maiores detalhes. Confira a documentação do seus SO para saber o valor máximo para esta variável. Tentativas de configurar `back_log` maior do que o limite de seu sistema operacional serão ineficazes.
- `basedir` O valor da opção `--basedir`.
- `bdb_cache_size` O buffer que é alocado para o cache de índice e registros de tabelas `BDB`. Se você não utiliza tabelas `BDB`, deve iniciar o `mysqld` com a opção `--skip-bdb` para evitar desperdício de memória para este cache.
- `bdb_log_buffer_size` O buffer que é alocado para o cache de índice e registros de tabelas `BDB`. Se você não utiliza tabelas `BDB`, deve configurá-la com 0 ou iniciar o `mysqld` com a opção `--skip-bdb` para evitar desperdício de memória para este cache.
- `bdb_home` O valor para a opção `--bdb-home`.
- `bdb_max_lock` O número máximo de bloqueios (1000 por padrão) que podem ser feitas em uma tabela BDB. Você deve ser aumentá-la se obter erros do tipo: `bdb: Lock table is out of available locks` ou `Got error 12 from ...` quando são necessárias longas transações ou quando o `mysqld` precisar examinar vários registros para calcular a pesquisa.
- `bdb_logdir` O valor da opção `--bdb-logdir`.
- `bdb_shared_data` Está ligada (`ON`) se você estiver utilizando `--bdb-shared-data`.
- `bdb_tmpdir` O valor da opção `--bdb-tmpdir`.
- `binlog_cache_size`. O tamanho do cache para armazenar instruções SQL para o log binário durante uma transação. Se você geralmente utiliza transações grandes, multi-instruções, você pode aumentar este valor para obter mais performance. Veja Seção 6.7.1 [COMMIT], Página 615.
- `bulk_insert_buffer_size` (era `myisam_bulk_insert_tree_size`) MyISAM usa uma cache especial em árvore para fazer inserções em bloco (isto é, `INSERT ... SELECT`, `INSERT ... VALUES (...), (...), ...`, e `LOAD DATA INFILE`) mais rápidos.

Esta variável limita o tamanho da árvore cache em bytes por thread. Definí-la com 0 desabilitará esta otimização **Nota**: esta cache só é usada quando é adicionado dados a uma tabela não vazia. O valor padrão é 8 MB.

- `character_set` O conjunto de caracteres padrão.
- `character_sets` Os conjuntos de caracteres suportados.
- `concurrent_inserts` Se `ON` (ligado, por padrão), o MySQL permitirá o uso de `INSERT` em tabelas `MyISAM` ao mesmo tempo em que são executadas consultas `SELECT`. Você pode desligar esta opção iniciando `mysqld` com `--safe` ou `--skip-new`.
- `connect_timeout` O número de segundos que o servidor `mysqld` espera para um pacote de conexão antes de responder com `Bad handshake`.
- `datadir` O valor da opção `--datadir`.
- `delay_key_write`

  Option for MyISAM tables. Can have one of the following values:

OFF	All `CREATE TABLE ... DELAYED_KEY_WRITE` são ignorados.
ON	(padrão) MySQL seguirá a opção `DELAY_KEY_WRITE` para `CREATE TABLE`.
ALL	Todas as novas tabelas abertas são tratadas como se fossem criadas com a opção `DELAY_KEY_WRITE`.

  Se `DELAY_KEY_WRITE` estiver habilitado, isto siginifica que o buffer de chaves das tabelas com esta opção não serão descarregadas a cada atualização do índice, mas somente quando a tabela é fechada. Isto irá aumentar bem a velocidade de escrita em chaves, mas você deve adicionar verificação automática de todas as tabelas com `myisamchk --fast --force` se você usá-lo.
- `delayed_insert_limit` Depois de inserir `delayed_insert_limit` registros, o agente que cuida de `INSERT DELAYED` ira conferir se exitem instruções `SELECT` pendentes. Se sim, ele permite a execução destas antes de continuar.
- `delayed_insert_timeout` Quanto tempo uma thread `INSERT DELAYED` deve esperar por instruções `INSERT` antes de terminar.
- `delayed_queue_size` Qual tamanho deve ser alocado para a fila (em linhas) para lidar com `INSERT DELAYED`. Se a fila encher, algum cliente que executar `INSERT DELAYED` irá esperar até existir espaço na fila novamente.
- `flush` É habilitado (`ON`) se você iniciar o MySQL com a opção `--flush`.
- `flush_time` Se esta variável for configurada com um valor diferente de zero, então a cada `flush_time` segundos todas tabelas serão fechadas (para economizar recursos e sincronizar dados com o disco). Recomendamos esta opção somente em sistemas com Win95, Win98 ou outros sistemas com poucos recursos.
- `ft_boolean_syntax` Lista de operadores suportados por `MATCH ... AGAINST(... IN BOOLEAN MODE)`. Veja Seção 6.8 [Pesquisa Fulltext], Página 619.
- `ft_min_word_len` O tamanho mínimo da palavra a ser incluída em um índice `FULLTEXT`. **Nota: índices `FULLTEXT` devem ser reconstruídos depois de alterar esta variável.** (Esta opção é nova para o MySQL 4.0.)
- `ft_max_word_len` O tamanho máximo da palavra a ser incluída em um índice `FULLTEXT`. **Nota: índices `FULLTEXT` devem ser reconstruídos depois de alterar esta variável.** (Esta opção é nova para o MySQL 4.0.)

- `ft_query_expansion_limit` Núnero de correspondências a usar para consulta de expansão (em `MATCH ... AGAINST (... WITH QUERY EXPANSION)`. (Esta opção é nova no MySQL 4.1.1)
- `ft_max_word_len_for_sort` O tamanho máximo da palavra a ser incluída em um índice `FULLTEXT` a ser usado no método rápido de recriação do índice em `REPAIR`, `CREATE INDEX`, ou `ALTER TABLE`. Palavras mais longas são inseridas de modo lento. A regra do dedão é a seguinte: com `ft_max_word_len_for_sort` aumentando, **MySQL** criará arquivos temporários maiores (tornando o processo lente, devido a E/S de disco), e colocará poucas chaves em um bloco ordenado (diminuindo a eficiência novamente). Quando `ft_max_word_len_for_sort` é muito pequeno, **MySQL** irá inserir várias palavras no índice de modo lento, mas pequenas palavras serão inseridas muito rapidamente.
- `ft_stopword_file` O arquivo do qual se lê a lista de palavras de parada para pesquisa fulltext. Todas as palavras do arquivo serão usadas; comentários **não** são seguidos. Por padrão, a lista já incluída de palavras de parada é a usada (como definido em '`myisam/ft_static.c`'). Definir este parâmetro com uma string vazia (`""`) disabilitaa o filtro de palavras de parada. **Nota: índices `FULLTEXT` devem ser reconstruídos depois de alterar esta variável.** (Esta opção é nova para o MySQL 4.0.)
- `have_innodb` `YES` if `mysqld` suporta tabelas InnoDB. `DISABLED` se `--skip-innodb` é usado.
- `have_bdb` `YES` se o `mysqld` suportar tabelas Berkeley DB. `DISABLED` se a opção `--skip-bdb` for usada.
- `have_raid` `YES` se o `mysqld` suportar a opção `RAID`.
- `have_openssl` `YES` se o `mysqld` suportar SSL (criptografia) no protocolo cliente/ servidor.
- `init_file` O nome do arquivo especificado com a opção `--init-file` quando você iniciar o servidor. Este é um arquivo das instruções SQL que você deseja que o servidor execute quando é iniciado.
- `interactive_timeout` O número de segundos que o servidor espera por atividade em uma conexão antes de fechá-la. Um cliente interativo é definido como um cliente que utiliza a opção `CLIENT_INTERACTIVE` para `mysql_real_connect()`. Veja também `wait_timeout`.
- `join_buffer_size` O tamanho do buffer que é utilizado para full joins (joins que não utilizam índices). O buffer é alocado uma vez para cada full join entre duas tabelas. Aumente este valor para obter um full join mais rápido quando a adição de índices não for possível. (Normalmente a melhor forma de obter joins rápidas é adicionar índices.)
- `key_buffer_size` Blocos de índices são buferizados e compartilhados por todas as threads. `key_buffer_size` é o tamanho do buffer utilizado para indexar blocos.

  Aumente-o para lidar melhor com os índices (para todas as leituras e escritas múltiplas) para o máximo possível 64M em uma máquina com 256M que executa, principalmente, o MySQL é bastante comum. Entretanto, se você deixar este valor muito grande (mais que 50% da sua memória total?) seu sistema pode iniciar a paginar e se tornar MUITO lento. Lembre-se que como o MySQL não utiliza cache de leitura de dados, será necessário deixar algum espaço para o cache do sistema de arquivos para o Sistema Operacional.

Você pode verificar a performance do buffer de chaves executando `SHOW STATUS` e examinar as variáveis `Key_read_requests`, `Key_reads`, `Key_write_requests` e `Key_writes`. A razão de `Key_reads/Key_read_request` deve normalmente ser `<` 0.01. O `Key_write/Key_write_requests` é normalmnte próximo de 1 se você estiver utilizando na maioria updates/deletes mas deve ser bem menor se você tender a fazer atualizações que afetam várias outras ao mesmo tempo ou se você estiver utilizando `DELAY_KEY_WRITE`. Veja Seção 4.6.8 [SHOW], Página 302.

Para obter ainda mais velocidade quando estiver gravando vários registros ao mesmo tempo, utilize `LOCK TABLES`. Veja Seção 6.7.5 [`LOCK TABLES`], Página 617.

- `language` A linguagem utilizada para mensagens de erro.
- `large_file_support` Se o `mysqld` foi compilado com opções para suporte a grandes arquivos.
- `locked_in_memory` Se o `mysqld` foi travado na memória com `--memlock`
- `log` Se o log de todas as consultas está habilitado.
- `log_update` Se o log de atualizações está habilitado.
- `log_bin` Se o log binários está habilitado.
- `log_slave_updates` Se as atualizações do slave devem ser logadas.
- `long_query_time` Se uma consulta demorar mais que isto (em segundos), o contador `Slow_queries` ser incrementado. Se você estiver utilizando `--log-slow-queries`, a consulta será logada ao arquivo de consultas lentas. Veja Seção 4.10.5 [Slow query log], Página 377. Este valor é medido em tempo real, não em tempo de CPU, assim uma consulta que pode estar pode estar abaixo do limiar de um sistema de carga leve pode estar acima do limiar de um sistema de carga pesada. Veja Seção 4.10.5 [Log de consultas lentas], Página 377.
- `lower_case_nome_tabelas` Se estiver configurado para 1, nomes de tabelas são armazenados em letras minúsculas no disco e nomes de tabelas serão caso-insensitivo. Na versão .0.2, esta opção também se aplica aos nomes de banco de dados. Na versão 4.1.1 esta opção também se aplcia a alias de tabelas. Veja Seção 6.1.3 [Caso sensitivo nos nomes], Página 473.
- `max_allowed_packet` O valor máximo de um pacote. O buffer de mensagens é iniciado por `net_buffer_length` bytes, mas pode crescer até `max_allowed_packet` bytes quando for necessário. Este valor por padrão é pequeno, para obter pacotes maiores (possivelmente errados). Você deve incrementar este valor se você estiver usando colunas `BLOB` grandes. Ele deve tão grande quanto o maior `BLOB` que você deseja utilizar. O protocol atual limita o `max_allowed_packet` à 16M no MySQL 3.23 e 1G no MySQL 4.0.
- `max_binlog_cache_size` Se uma transação multi-instruções necessitar de mais que este montante de memória, será obtido o erro "Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage" ("Transação multi-instruções necessita mais que 'max_binlog_cache_size' bytes de armazenamento").
- `max_binlog_size` Disponível a partir da 3.23.33. Se uma escrita ao log binário (replicação) exceder o valor fornecido, rotacione os logs. Você não pode configurá-lo para menos de 4096 bytes (1024 na versão do MySQL anteriores a 4.0.14), ou mais que 1 GB. O valor padrão é 1 GB. Nota se você estiver usando transações: uma transação

é escrita em um bloco no arquivo de log binário, já que ele nunca é separado entre diversos logs binários. Desta forma, se você tiver grandes transações, você pode ter logs binários maioores que `max_binlog_size`. Se `max_relay_log_size` (disponível a partir do MySQL 4.0.14) é 0, então `max_binlog_size` se aplicará bem aos relay logs.

- `max_connections` O Número de clientes simultâneos permitidos. Aumentar este valor aumenta o número de descritores de arquivos que o `mysqld` necessita. Veja abaixo os comentários sobre os limites de descritores de arquivos. Veja Seção A.2.6 [Muitas conexões], Página 916.
- `max_connect_errors` Se houver mais que este número de conexões interrompidas a partir de uma máquina está máquina terá as próximas conexões bloqueadas. Você pode desbloquar uma máquina com o comadno `FLUSH HOSTS`.
- `max_delayed_threads` Não inicie mais do que este número de threads para lidar com instruções `INSERT DELAYED`. Se você tentar inserir dados em uma nova tabela depois que todas as threads `INSERT DELAYED` estiverem em uso, o registro será inserido como se o atributo `DELAYED` não fosse especificado. Se você configurá-lo com 0, o MySQL nunca criará uma thread max_delayed.
- `max_heap_table_size` Esta variável define o tamanho máximo que uma tabela `HEAP` criada pode ter. O valor da variável é usado para calcular um valor `MAX_ROWS` da tabela `HEAP`. A definição desta variável não tem nenhum efeito sobre qualquet tabela `HEAP` existente, a memos que a tabela seja recriada com uma instrução como `CREATE TABLE` ou `TRUNCATE TABLE`, ou alterada com `ALTER TABLE`.
- `max_join_size` Joins que provavelmente forem ler mais que `max_join_size` registros retornam um erro. Configure este valor se os seus usuários tendem a realizar joins que não possuem uma cláusula `WHERE`, que tomam muito tempo, e retornam milhões de registros.
- `max_relay_log_size` Disponível a partir da versão 4.0.14. Se uma escrita ao relay log (um tipo de log usado por slaves de replicação, veja Seção 4.11.3 [Replication Implementation Details], Página 381) exceder o valor dado, rotacione o relay log. Esta variável lhe permite colocar diferentes restrições de tamanho no relay logs e logs binários. No entanto, configurar a variável com 0 fará o MySQL usar `max_binlog_size` tanto para o log binário quanto para o relay logs. Você tem que configurar `max_relay_log_size` com 0 ou mais de 4096, e menos que 1 GB. O padrão é 0.
- `max_seeks_for_key` Limite do número máximo de buscas ao procurar linhas com base em uma chave. O otimizador MySQL assumirá que quando pesquisar por linhas correspondentes em uma tabela através da varredura da chave, não faremos mais que este número de busca de chave independente da cardinalidade da chave. Configurando este parâmetro com um valor baixo (100 ?) você pode forçar o MySQL a preferir chaves em vez de varrer a tabela.
- `max_sort_length` O número de bytes utilizados para ordenar valores `BLOB` ou `TEXT` (somente os primeiros `max_sort_lenght` bytes de cada valor são usados; o resto é ignorado).
- `max_user_connections` O valor máximo de conexões ativas para um único usuário (0 = sem limite).
- `max_tmp_tables` (Esta opção ainda não faz nada.) Número máximo de tabelas temporárias que um cliente pode manter abertas ao mesmo tempo.

- `max_write_lock_count` Depois desta quantidade de bloqueios de escrita, permite que alguns bloqueios de leitura sejam executados.
- `myisam_recover_options` O valor da opção `--myisam-recover`.
- `myisam_sort_buffer_size` O buffer que é alocado ao ordenar o índice quando estiver fazendo um `REPAIR` ou estiver criando índices com `CREATE INDEX` ou `ALTER TABLE`.
- `myisam_max_extra_sort_file_size`. Se a criação do arquivo temporário para criação rápida de índices fosse este valor maior que quando for usado o cache de chaves, de preferência ao método de cache de chaves. Isto é usado principalmente para forçar que longas chaves de caracteres em tabelas grandes usem o método de cache de chaves mais lenta para criar o índice. **NOTE** que este parâmetro é fornecido em megabytes!
- `myisam_repair_threads`. Se este valor é maior que um, durante o processo `reparo por ordenação` os índices de tabels MyISAM serão criados em paralelo - cada índice em sua própria thread. `Nota` reparos com multi-threads está ainda sob código de qualidade **alpha**.
- `myisam_max_sort_file_size` O tamanho máximo do arquivo temporário que é permitido ao MySQL usar enquanto recria os índices (durante `REPAIR`, `ALTER TABLE` ou `LOAD DATA INFILE`). Se o tamanho do arquivo for maior que isto, o índice será criado através do cache de chaves (que é mais lento). **NOTE** que este parâmetro é fornecido em megabytes antes da versão 4.0.3 e em bytes a partir desta versão.
- `net_buffer_length` O buffer de comunicações é configurado para este tamanho entre queries. Isto não deve ser alterado normalmente, mas se você tem muito pouca memória, pode configurá-lo para o tamanho esperado de uma consulta. (Isto é, o tamanho experado das instruções SQL enviadas pelos clientes. Se as instruções excederem este valor, o buffer é aumentado automaticamente, até `max_allowed_packet` bytes.)
- `net_read_timeout` Número de segundos para esperar por mais dados de uma conexão antes de abortar a leitura. Perceba que quando nós não esperamos dados de uma conexão, o tempo máximo de espera é definido pelo `write_timeout`. Veja também `slave_read_timeout`.
- `net_retry_count` Se uma leitura na porta de comunicações for interrompida, tente novamente `net_retry_count` vezes antes de parar. Este valor deve ser bem alto no `FreeBSD` já que interrupções internas são enviadas para todas as threads.
- `net_write_timeout` Número de segundos para esperar pela escrita de um bloco em uma conexão antes de abortar a escrita.
- `open_files_limit` Número de arquivos que o sistema permite que o mysqld abra. Este é o valor real dado para o sistema e pode ser diferente do valor que você passa ao mysqld como parâmetro de inicialização. Ele é 0 em sistemas onde o MySQL não pode alterar o número de arquivos abertos.
- `pid_file` O valor da opção `--pid-file`.
- `port` O valor da opcao `--port`.
- `protocol_version` A versão do protocolo usada pelo servidor MySQL.
- `range_alloc_block_size` Tamanho dos blocos que são alocados ao se fazer uma otimização da faixa.

- `read_buffer_size` (era `record_buffer`) Cada thread que faz uma leitura sequencial aloca um buffer deste tamanho para cada tabela lida. Se você fizer várias leituras sequenciais, você pode desejar aumentar este valor.
- `read_rnd_buffer_ae` (era `record_rnd_buffer`) Ao ler registros na ordem depois de uma ordenação, os registros são lidos através deste buffer para evitar pesquisas em disco. Pode melhorar bastante o `ORDER BY` se configurado com um valor alto. Como esta é uma variável específica da thread, não se pode definí-la globalmente, mas apenas alterá-la ao executar alguma consulta específica grande.
- `query_alloc_block_size` Tamanho dos blocos de alocação de memória que são alocados para objetos criados durante a análise e execução da consulta. Se você tiver problemas com fragmentação de memória ele pode ajudar a aumentar isto um pouco.
- `query_cache_limit` Não armazena resultados que são maiores que esta variável. (Padrão 1M).
- `query_cache_size` A memória alocada para armazenar resultados de consultas antigas. Se 0, a cache de consulta é desabilitada (padrãot).
- `query_cache_type` Pode ser configurado com (somente numérico)

Valor	Alias	Comentário
0	OFF	Não armazena ou recupera resultados
1	ON	Armazena todos os resultados exceto consultas `SELECT SQL_NO_CACHE ...`
2	DEMAND	Armazena apenas consultas `SELECT SQL_CACHE ...`

- `query_prealloc_size` Buffer persistente para análise e execução da consulta. Não é liberado entre consultas. Em teoria, tornando-o "grande o suficiente" você pode fazer o MySQL executar consultas sem ter que fazer uma única chamada `malloc`.
- `safe_show_database` Não exibe bancos de dados nos quais o usuário não tem nenhum privilégios. Isto pode melhorar a segurança se você se preocupa com o fato das pessoas estarem aptas a ver quais bancos de dados outros usuários possuem. Veja também `skip_show_databases`.
- `server_id` O valor da opção `--server-id`.
- `skip_locking` Está desligado (OFF) se o `mysqld` usar bloqueio externo.
- `skip_networking` Está ligado (ON) se somente permitimos conexões locais (socket).
- `skip_show_databases` Isto previne usuários de fazerem `SHOW DATABASES` se eles não possuirem o privilégio `PROCESS_PRIV`. Isto pode aumentar a segurança se você se preocupa com o fato das pessoas poderem ver quais bancos de dados outros usuários possuem. Veja também `safe_show_databases`.
- `slave_net_timeout` Número de segundos para esperar por mais dados de uma conexão de master/slave antes de abortar a leitura.
- `slow_launch_time` Se a criação de threads demorar mais que este valor (em segundos), o contador `Slow_launch_threads` será incrementado.
- `socket` O socket Unix utilizado pelo servidor.
- `sort_buffer` Cada thread que precisar fazer uma ordenação aloca um buffer deste tamanho. Aumente este valor para operações `ORDER BY` ou `GROUP BY` mais rápidas. Veja Seção A.4.4 [Arquivos temporários], Página 929.

- `table_cache` O número de tabelas abertas para todas as threads. Aumentar este valor aumenta o número de descritores de arquivos que o `mysql` necessita. O MySQL precisa de dois descritores de arquivos para cada tabela única aberta. Veja abaixo os comentaários sobre os limites do descritor de arquivos. Você pode conferir se necessita aumentar o cache de tabela conferindo a variável `Opened_tables`. Veja Seção 4.6.8.3 [`Tabelas Abertas`], Página 305. Se esta variável for grande e você não faz muitos `FLUSH TABLES` (que apenas força todas as tabelas a serem fechadas e reabertas), então você deve aumentar o valor desta variável.

  Para informações sobre como o cache de tabelas funciona, veja Seção 5.4.7 [Table cache], Página 451.
- `table_type` O tipo padrão de tabelas.
- `thread_cache_size` Quantas threads devem ser mantidas em cache para reutilização. Quando um cliente desconecta, as threads dos clientes são colocadas no cache se não existir mais de `thread_cache_size` threads que antes. Todas novas threads serão obtidas primeiramente do cache, e só quando o cache estiver vazio uma nova thread é criada. Esta variável pode ser aumentada para melhorar a performance se você tiver várias conexões novas. (Normalmente isto não dá uma melhora de performance notável se você possuir uma boa implementação de threads.) Examinando as diferenças entre `Connections` e `Threads_create` (veja Seção 4.6.8.3 [`SHOW STATUS`], Página 305 para maiores detalhes) pode ser visto o quão eficente é o cache de threads atual.
- `thread_concurrency` No Solaris, `mysqld` irá chamar `thr_setconcurrency()` com este valor. `thdr_setconcurrency()` permite que a aplicação forneça ao sistema de threads uma dica com o número desejado de threads que devem ser executados ao mesmo tempo.
- `thread_stack` O tamanho da pilha para cada thread. Vários dos limites detectados pelo teste `crash-me` são dependentes deste valor. O padrão é grande o suficiente para operações normais. Veja Seção 5.1.4 [Benchmarks do MySQL], Página 421.
- `timezone` O fuzo horário para este servidor.
- `tmp_table_size` Se uma tabela temporária em memória exceder este tamanho, o MySQL irá a convertê-la automaticamente para uma tabela `MyISAM` em disco. Aumente o valor de `tmp_table_size` se você fizer várias consultas `GROUP BY` avançadas e você tiver muita memória.
- `tmpdir` O diretório utilizado para arquivos temporários e tabelas temporárias. A partir do MySQL 4.1, ele pode ser definido com uma lista de caminhos separados por dois pontos (`:`) (ponto e vírgula (`;` no Windows). Eles serão usados de modo robin-round. Este recurso pode ser usado para dividir a craga entre diversos discos físicos.
- `transaction_alloc_block_size` Tamanho dos blocos de alocação de memória que são alocados para consultas de armazenamento que são parte de uma transação que está para ser armazenada no log binário ao se fazer um commit.
- `transaction_prealloc_block_size` Buffer persistente para `transaction_alloc_blocks` que não é liberado entre as consultas. Tornando-o "grande o suficiente" para caber todas as consulta em uma transação comum você pode evitar muitas chamadas `malloc`.
- `version` O número da versão do servidor.

- `wait_timeout` O número de segundos que o servidor espera pela atividade em uma conexão antes de fechá-la. Veja também `interactive_timeout`.

  Na inicialização da thread, `SESSION.WAIT_TIMEOUT` é inicializado por `GLOBAL.WAIT_TIMEOUT` ou `GLOBAL.INTERACTIVE_TIMEOUT` dependendo do tipo do cliente (como definido pela opção de conexão `CLIENT_INTERACTIVE`). Veja também `interactive_timeout`.

A seção do manual que descreve o ajuste do MySQL contém algumas informações de como sintonizar as variáveis acima. Veja Seção 5.5.2 [Parâmetros do servidor], Página 454.

4.6.8.5 SHOW [BDB] LOGS

`SHOW LOGS` exibe estatísticas sobre os arquivos log existentes. Atualmente ele só exibe informações sobre arquivos de log Berkeley DB, assim um alias para ele (disponível a partir do MySQL 4.1.1) é `SHOW BDB LOGS`.

- `File` exibe o caminho completo para o arquivo de log
- `Type` exibe o tipo do arquivo log (`BDB` para arquivos de log Berkeley DB).
- `Status` exibe o status do arquivo log (`FREE` se o arquivo pode ser removido, ou `IN USE` se o arquivo é necessário para o subsistema de transações)

4.6.8.6 SHOW PROCESSLIST

`SHOW [FULL] PROCESSLIST` exibe quais threads estão em execução. Esta informação também pode ser obtida com o comando `mysqladmin processlist`. Se você possuir o privilégio `SUPER`, poderá ver todas as threads. Senão só é possível ver as próprias threads. Veja Seção 4.6.7 [KILL], Página 301. Se você não utiliza a opção `FULL`, então somente os primeiros 100 caracteres de cada query serião exibidos.

A partir da versão 4.0.12, o MySQL informa o nome de maquina para conexões TCP/IP no formato `nome_maquina:client_port` para tornar mais fácil de se encontrar qual cliente está fazendo o que.

Este comando é muito útil caso você obtenha a mensagem de erro 'too many connections' e deseja saber o que está ocorrendo. O MySQL reserva uma conexão extra por cliente com o privilégio `SUPER` para garantir que você sempre consiga logar e conferir o sistema (assumindo que este privilégio não foi concedido para todos os usuários).

Alguns estados normalmente vistos em `mysqladmin processlist`

- `Checking table` A thread está realizando verificação [automática] da tabela.
- `Closing tables` Signiifica que a thread está descarregando os dados alterados na tabela para o disco e fechando as tabelas usadas. Isto deve ser uma operação rápida. Se não, você deve verificar se o seu disco não está cheio ou que o disco não está com sobrecarga.
- `Connect Out` Slave está conectando ao master.
- `Copying to tmp table on disk` O resultado temporário foi maior que `tmp_table_size` e a thread agora está alterando a tabela temporária na memória para o disco para economizar memória.
- `Creating tmp table` A thread está criando uma tabela temporária para guardar uma parte do resultado para a consulta.

- `deleting from main table` Ao executar a primeira parte de um delete multi-tabela e estamos deletando apenas da primeira tabela.
- `deleting from reference tables` Ao executar a segunda parte de um delete multi-tabela e estamos deletando o registros correspondentes em outras tabelas.
- `Flushing tables` A thread está executando `FLUSH TABLES` e está esperando que todas as threads fechem as suas tabelas.
- `Killed` Alguém enviou um sinal para matar a thread e ela deve abortar a próxima vez que ele verificar o parâmetro kill. O parâmetro é verificado em cada loop maior no MySQL, mas em alguns casos ainda pode levar um tempo curto para a thread morrer. Se a thread está bloqueada par outra thread, a finalização terá efeito assim que as outras threads liberarem o bloqueio.
- `Sending data` A thread está processando registros para uma instrução `SELECT` e também está enviando dados ao cliente.
- `Sorting for group` A thread está fazendo uma ordenação para satisfazer a um `GROUP BY`.
- `Sorting for order` A thread está fazendo uma ordenação para satisfazer a um `ORDER BY`.
- `Opening tables` Isto simplesmente significa que a thread está tentando abrir uma tabela. Este deve ser um procedimento muito rápido, a menos que algo previna da abertura. Por exemplo um `ALTER TABLE` ou um `LOCK TABLE` pode prvenir a abertura de uma tabela até que o comando esteja finalizado.
- `Removing duplicates` A consulta estava usando `SELECT DISTINCT` de tal modo que o MySQL não podia otimizar o distinct em um estagio anterior. Por isto o MySQL fez um estágio extra para remover todos os registros duplicados antes de enviar o resultado ao cliente.
- `Reopen table` A thread obteve um lock para a tabela, mas notificou após o lock que a estrutura da tabela alterou. Ela liberou o lock, fechou a tabela e agora está tentando reabrí-la.
- `Repair by sorting` O código de reparação está utilizando ordenamento para recriar os índices.
- `Repair with keycache` O código de reparação está usando a criação de chaves uma a uma através da cache de chaves. Isto é muito mais lento que `Repair by sorting`.
- `Searching rows for update` A thread esta fazendo uma primeira fase pra encontrar todos os registros coincidentes antes de atualizá-los. Isto deve ser feito se o `UPDATE` está alterando o índice usado para encontrar os registros envolvidos.
- `Sleeping` A thread está esperando que o cliente envie um novo comando a ela.
- `System lock` A thread está esperando um lock de sistema externo para a tabela. Se você não está usando múltiplos servidores mysqld que estão acessando a mesma tabela, você pode desabilitar o lock de sistema com a opção `--skip-external-locking`.
- `Upgrading lock` O manipulador de `INSERT DELAYED` está tentando obter um lock para inserir registros na tabela.
- `Updating` A thread está procurando por registros para atualizá-los.
- `User Lock` A thread está esperando um `GET_LOCK()`.

- `Waiting for tables` A thread recebeu uma notificação que a estrutura de uma tabela foi alterada e ela precisa reabrir a tabela para receber a nova estrutura. Para poder reabrir a tabela ela deve esperar até que todas a outras threads tenham fechado a tabela em questão.

  A notificação acontece se outra thread usou `FLUSH TABLES` ou um dos seguintes comando na tabela em questão: `FLUSH TABLES nome_tabela`, `ALTER TABLE`, `RENAME TABLE`, `REPAIR TABLE`, `ANALYZE TABLE` ou `OPTIMIZE TABLE`.
- `waiting for handler insert` O manipulador do `INSERT DELAYED` processou todas as inserções e está esperado por outras.

A maioria dos estados são operações muito rápidas. Se a thread permanecer em qualquer destes estados por muitos segundos, pode haver um problema que precisa ser investigado.

Existem outros estados que não são mencionados anteriormente, mas a maioia deles só são úteis para encontrar erros no `mysqld`.

4.6.8.7 SHOW GRANTS

`SHOW GRANTS FOR usuário` lista os comandos concedidos que devem ser usados para duplicar os direitos de um usuário.

```
mysql> SHOW GRANTS FOR root@localhost;
+---------------------------------------------------------------------+
| Grants for root@localhost                                           |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+---------------------------------------------------------------------+
```

Para listar as permissões da sessão atual pode-se usar a função `CURRENT_USER()` (nova na versão 4.0.6) para descobrir com qual usuário a sessão foi autenticada. Veja Seção 6.3.6.2 [`CURRENT_USER()`], Página 547.

4.6.8.8 SHOW CREATE TABLE

Exibe uma instrução `CREATE TABLE` que irá criar a seguinte tabela:

```
mysql> SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE t (
  id INT(11) default NULL auto_increment,
  s char(60) default NULL,
  PRIMARY KEY (id)
) TYPE=MyISAM
```

`SHOW CREATE TABLE` cita os nomes de colunas e tabelas de acordo com o valor da opção `SQL_QUOTE_SHOW_CREATE`. Seção 5.5.6 [`SET OPTION SQL_QUOTE_SHOW_CREATE`], Página 460.

4.6.8.9 SHOW WARNINGS | ERRORS

```
SHOW WARNINGS [LIMIT row_count]
SHOW ERRORS [LIMIT row_count]
```

Este comando é implementado no MySQL 4.1.0.

Ele mostra os erros,a visos e notas recebidos para o último comando. Os erros/avisos são reiniciados para cada comando que utiliza uma tabela.

O servidor MySQL envia de volta o número total de avisos e erros que você recebe para o último comando; Isto pode ser retornado chamando `mysql_warning_count()`.

Até as mensagens `max_error_count` são armazenadas (variáveis global e específicas da thread).

Você pode recuperar o número de erros de `@error_count` e avisos de `@warning_count`.

`SHOW WARNINGS` mostra todos os erros, avisos e notas que você recebeu para o último comando enquanto `SHOW ERRORS` lhe mostra apenas o erro.

```
mysql> DROP TABLE IF EXISTS no_such_table;
mysql> SHOW WARNINGS;

+-------+------+-------------------------------+
| Level | Code | Message                       |
+-------+------+-------------------------------+
| Note  | 1051 | Unknown table 'no_such_table' |
+-------+------+-------------------------------+
```

Note que no MySQL 4.1.0 apenas adicionamos a estrutura para avisos e poucos comandos MySQL ainda geraram avisos. A versão 4.1.1 suporta todos os tipos de avisos para `LOAD DATA INFILE` e instruções DML tais como os comandos `INSERT`, `UPDATE` e `ALTER`.

Por exemplo, aqui está um caso simple que produz avisos de conversão para instruções de inserção.

```
mysql> create table t1(a tinyint NOT NULL, b char(4));
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t1 values(10,'mysql'),(NULL,'test'),(300,'open source');
Query OK, 3 rows affected, 4 warnings (0.15 sec)
Records: 3  Duplicates: 0  Warnings: 4

mysql> show warnings;
+---------+------+-----------------------------------------------------------------+
| Level   | Code | Message                                                         |
+---------+------+-----------------------------------------------------------------+
| Warning | 1263 | Data truncated for column 'b' at row 1                          |
| Warning | 1261 | Data truncated, NULL supplied to NOT NULL column 'a' at row 2 |
| Warning | 1262 | Data truncated, out of range for column 'a' at row 3            |
| Warning | 1263 | Data truncated for column 'b' at row 3                          |
+---------+------+-----------------------------------------------------------------+
4 rows in set (0.00 sec)
```

O número máximo de avisos pode ser específicado usando a variável do servidor `'max_error_count'`, `SET max_error_count=[count]`; Por padrão é 64. No caso de avisos desabilitados, simplesmente zere esta variável. No caso de `max_error_count` ser 0, então o contador de avisos ainda representa quantos avisos ocorreram, mas nenhuma das mensagens são armazenadas.

Por exemplo, considere o seguinte instrução de tabela `ALTER` para o exemplo acima, o qual retorna apenas um mensagem de aviso embora o total de avisos seja 3, ao definir max_error_count=1.

```
mysql> show variables like 'max_error_count';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_error_count | 64    |
+-----------------+-------+
1 row in set (0.00 sec)

mysql> set max_error_count=1;
Query OK, 0 rows affected (0.00 sec)

mysql> alter table t1 modify b char;
Query OK, 3 rows affected, 3 warnings (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 3

mysql> show warnings;
+---------+------+---------------------------------------+
| Level   | Code | Message                               |
+---------+------+---------------------------------------+
| Warning | 1263 | Data truncated for column 'b' at row 1 |
+---------+------+---------------------------------------+
1 row in set (0.00 sec)

mysql>
```

4.6.8.10 SHOW TABLE TYPES

```
SHOW TABLE TYPES
```

Este comando é implementado no MySQL 4.1.0.

`SHOW TABLE TYPES` lhe mostra a informação de status sobre o tipo de tabela. Isto é particulamente útil para verificar se um tipo de tabela é suportado; ou para ver qual é o tipo de tabela padrão.

```
mysql> SHOW TABLE TYPES;

+--------+---------+----------------------------------------------------------------+
| Type   | Support | Comment                                                        |
+--------+---------+----------------------------------------------------------------+
| MyISAM | DEFAULT | Default type from 3.23 with great performance                  |
```

```
| HEAP   | YES     | Hash based, stored in memory, useful for temporary tables |
| MERGE  | YES     | Collection of identical MyISAM tables                     |
| ISAM   | YES     | Obsolete table type; Is replaced by MyISAM                |
| InnoDB | YES     | Supports transactions, row-level locking and foreign keys |
| BDB    | NO      | Supports transactions and page-level locking              |
+--------+---------+-----------------------------------------------------------+
6 rows in set (0.00 sec)
```

A opção 'Support' `DEFAULT` indica se um tipo de tabela particular é é suportado, e qual é o tipo padrão. Se o servidor é iniciado com `--default-table-type=InnoDB`, então o campo 'Support' do InnoDB terá o valor `DEFAULT`.

4.6.8.11 SHOW PRIVILEGES

```
SHOW PRIVILEGES
```

Este comando é implementado no MySQL 4.1.0.

`SHOW PRIVILEGES` mostra a lista de privilégios de sistema o servidor MySQL suporta.

```
mysql> show privileges;
+------------+--------------------------+-----------------------------------------
| Privilege  | Context                  | Comment
+------------+--------------------------+-----------------------------------------
| Select     | Tables                   | To retrieve rows from table
| Insert     | Tables                   | To insert data into tables
| Update     | Tables                   | To update existing rows
| Delete     | Tables                   | To delete existing rows
| Index      | Tables                   | To create or drop indexes
| Alter      | Tables                   | To alter the table
| Create     | Databases,Tables,Indexes | To create new databases and tables
| Drop       | Databases,Tables         | To drop databases and tables
| Grant      | Databases,Tables         | To give to other users those privileges y
| References | Databases,Tables         | To have references on tables
| Reload     | Server Admin             | To reload or refresh tables, logs and pri
| Shutdown   | Server Admin             | To shutdown the server
| Process    | Server Admin             | To view the plain text of currently execu
| File       | File access on server    | To read and write files on the server
+------------+--------------------------+-----------------------------------------
14 rows in set (0.00 sec)
```

4.7 Localização do MySQL e Utilização Internacional

4.7.1 O Conjunto de Caracteres Utilizado para Dados e Ordenação

Por padrão, o MySQL utiliza o conjunto de caracteres ISO-8859-1 (Latin1) com ordenação de acordo com o sueco/finlandês. Este também é o conjunto de caracteres aplicável nos EUA e oeste da Europa.

Todos os binários padrões do MySQL são compilados com `--with-extra-charsets=complex`. Isto adicionará código a todos os programas padrões para estarem aptos a lidar com o conjuntos de caracteres `latin1` e todos os multi-byte no binário. Outros conjuntos de caracteres serão carregados de um arquivo de definições de conjuntos de caracteres quando necessários.

O conjunto de caracteres determina quais são os caracteres permitidos em nomes e qual a forma de ordenação por cláusulas `ORDER BY` e `GROUP BY` da instrução `SELECT`.

Você pode alterar o conjunto de caracteres com a opção `--default-character-set` na inicialização do servidor. Os conjuntos de caracteres disponíveis dependem dos parâmetros `--with-charset=charset` e `--with-extra-charset= list-of-charset | complex | all | none` e os arquivos de configurações de conjuntos de caracteres listados em '`SHAREDIR/charsets/ Index`'. Veja Seção 2.3.3 [Opções de configurações], Página 98.

Se o conjunto de caracteres for alterado durante a execução do MySQL (que também pode alterar a ordenação), deve-se executar o `Omyisamchk -r -q --set-character-set=charset` em todas as tabelas. De outra forma seus índices podem não ser ordenados corretamente.

Quando um cliente conecta a um servidor MySQL, o servidor envia o conjunto de caracteres padrão em uso ao cliente. O cliente irá alternar para o uso deste conjunto de caracteres nesta conexão.

Deve ser utilizado `mysql_real_escape_string()` quando desejar ignorar seguências de caracteres em uma consulta SQL. `mysql_real_escape_string()` é identico à antiga função `mysql_espace_string()`, exceto pelo fato de usar a manipulador de conexão MySQL como o primeiro parâmetro.

Se o cliente for compilado com o caminho diferente daquele onde o servidor está instalado e o usuário que configurou o MySQL não incluiu todos os conjuntos de caracteres no binários do MySQL, deve ser especificado para o cliente onde ele pode encontrar os conjuntos de caracteres adcicionais que serão necessários se o servidor executar com um conjunto de caracteres diferente do cliente.

Isto pode ser especificado colocando em um arquivo de opções do MySQL:

```
[client]
character-sets-dir=/usr/local/mysql/share/mysql/charsets
```

onde o caminho aponta para onde os conjuntos de caracteres dinâmicos do MySQL são armazenados.

Pode-se forçar o cliente a usar conjuntos de caracteres específicos especificando:

```
[client]
default-character-set=nome-conjunto-caracteres
```

mas normalmente isto nunca será necessário.

4.7.1.1 German character set

Para se fazer ordenação em Alemão, você deve iniciar o `mysqld` com `--default-character-set=latin1_de`. Isto lhe dará as seguintes caracteristicas.

Ao ordenar e comparar strings, o seguinte mapeamento é feito na string antes de fazer a comparação:

```
ä ->  ae
ö  ->  oe
ü  ->  ue
ß  ->  ss
```

Todos os caracteres acentuados, são convertidos para suas contra partes sem acentos e em letras maiúsculas. Todas as letras são convertidas para maiúsculas.

Ao compara strings com `LIKE` o mapeamento de caracteres de um -> dois não é feito. Todas as letras são convertidas para maiúsculas. Acentos são removidos para todas as letras exceto: `Ü`, `ü`, `Ö`, `ö`, `Ä` e `ä`.

4.7.2 Mensagens de Erros em Outras Línguas

`mysqld` pode exibir mensagens de erros nas seguintes línguas: Tcheco, Dinamarquês, Holandês, Inglês (padrão), Estonian, Francês, Alemão, Grego, Húngaro, Italiano, Japonês, Koreano, Norueguês, Norueguês-ny, Polonês, Português, Romeno, Russo, Eslovaco, Espanhol e Sueco.

Para iniciar o `mysqld` com uma língua particular, use uma das opções: `--language=língua` ou `-L língua` . Por exemplo:

```
shell> mysqld --language=swedish
```

ou:

```
shell> mysqld --language=/usr/local/share/swedish
```

Perceba que todos as línguas são especificados em minúsculas.

Os arquivos de linguagens estão localizados (por padrão) em '*mysql_base_dir*/share/*LANGUAGE*/'.

Para atualizar o arquivo com mensagens de erros, deve-se editar o arquivo ' `errmsg.txt`' e executar o seguinte comando para gerar o arquivo '`errmsg.sys`':

```
shell> comp_err errmsg.txt errmsg.sys
```

Se você atualizar o MySQL para uma versão mais nova, lembre-se de repetir as alterações no novo arquivo '`errmsg.txt`'.

4.7.3 Adicionando um Novo Conjunto de Caracteres

Para adicionar outro conjunto de caracteres ao MySQL, utilize o seguinte procedimento.

Decida se o conjunto é simples ou complexo. Se o conjunto de caracteres não necessitar do uso de rotinas especiais de classificação de strings para ordenação e também não necessitar de suporte à caracteres multi-byte, será simples. Se ele necessitar de alguns destes recursos, será complexo.

Por exemplo, `latin1` e `danish` são conjuntos simples de caracteres enquanto `big5` ou `czech` são conjuntos de caracteres complexos.

Na seguinte seção, assumimos que você nomeou seu conjunto de caracteres como `MYSET`.

Para um conjunto de caracteres simples use o seguinte:

1. Adicione MYSET para o final do arquivo '`sql/share/charsets/Index`' Associe um número único ao mesmo.

2. Crie o arquivo '`sql/share/charsets/MYSET.conf`'. (O arquivo '`sql/share/charsets/latin1.conf`' pode ser utilizado como base para isto).

   A sintaxe para o arquivo é muito simples:

   - Comentários iniciam com um caractere '#' e continuam até o fim da linha.
   - Palavras são separadas por quantidades arbitrárias de espaços em brancos.
   - Ao definir o conjunto de caracteres, cada palavra deve ser um número no formato hexadecimal
   - O vetor `ctype` obtêm as primeiras 257 palavras. Os vetores `to_lower`, `to_upper` e `sort_order` obtêm, cada um, as 256 palavras seguintes.

   Veja Seção 4.7.4 [Vetor de caracteres], Página 329.
3. Adicione o nome do conjunto de caracteres às listas `CHARSETS_AVAILABLE` e `COMPILED_CHARSETS` no `configure.in`.
4. Reconfigure, recompile e teste.

Para um conjunto de caracteres complexo faça o seguinte:

1. Crie o arquivo '`strings/ctype-MYSET.c`' na distribuição fonte do MYSQL.
2. Adicione MYSET ao final do arquivo '`sql/share/charsets/Index`'. Associe um número único a ele.
3. Procure por um dos arquivos '`ctype-*.c`' existentes para ver o que precisa ser definido, por exemplo '`strings/ctype-big5.c`'. Perceba que os vetores no seu arquivo deve ter nomes como `ctype_MYSET`, `to_lower_MYSET` e etc. Isto corresponde aos arrays no conjunto simples de caracteres - Seção 4.7.4 [Vetor de caracteres], Página 329 - para um conjunto de caracteres complexo.
4. Próximo ao topo do arquivo, coloque um comentário especial como este:

   ```
   /*
    * This comment is parsed by configure to create ctype.c,
    * so don't change it unless you know what you are doing.
    *
    * .configure. number_MYSET=MYNUMBER
    * .configure. strxfrm_multiply_MYSET=N
    * .configure. mbmaxlen_MYSET=N
    */
   ```

   O programa `configure` utiliza este comentário para incluir o conjunto de caracteres na biblioteca MySQL automaticamente.

   As linhas strxfrm_multiply e mbmaxlen serão explicadas nas próximas seções. Só as inclua se você precisar de funções de ordenação de strings ou das funções de conjuntos de caracteres multi-byte, respectivamente.
5. Você deve então criar algumas das seguintes funções:

   - `my_strncoll_MYSET()`
   - `my_strcoll_MYSET()`
   - `my_strxfrm_MYSET()`
   - `my_like_range_MYSET()`

   Veja Seção 4.7.5 [Ordenação de caracteres], Página 329.

6. Adicione o nome do conjunto de caracteres às listas `CHARSETS_AVAILABLE` e `COMPILED_CHARSETS` no `configure.in`.
7. Reconfigure, recompile e teste.

O arquivo '`sql/share/charsets/README`' fornece algumas instruções a mais.

Se você desejar ter o seu conjunto de caracteres incluído na distribuição MySQL, envie um email com um patch para a lista de email "internals" do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33.

4.7.4 Os Vetores de Definições de Caracteres

`to_lower[]` e `to_upper[]` são vetores simples que definemm os caracteres minúsculos e maísculos correspondentes a cada membro do conjunto de caracteres. Por exemplo:

```
to_lower['A'] deve conter 'a'
to_upper['a'] deve conter 'A'
```

`sort_order[]` é um mapa indicando como os caracteres devem ser ordenados para propósitos de comparação e ordenação. Para vários conjuntos de caracteres, isto é o mesmo que `to_upper[]` (que significa ordenar em caso insensitivo). O MySQL ordenará caracteres baseado no valor de `sort_order[caractere]`. Para regras mais complicadas de ordenação, veja a discussão sobre ordenação de string abaixo. Veja Seção 4.7.5 [Ordenação de strings], Página 329.

`ctype[]` é um vetor com valores binários, com um elemento para cada caracter. (Note que `to_lower[]`, `to_upper[]` e `sort_order[]` são indexados pelo valor do caracter, mas o `ctype[]` é indexado pelo valor do caracter + 1. Este é um antigo legado para tratamento de EOF.)

Pode-se encontrar as seguintes máscaras binárias de definições em '`m_ctype.h`':

```
#define _U      01      /* Maísculo */
#define _L      02      /* Minúsculo */
#define _N      04      /* Numeral (digito) */
#define _S      010     /* Caractere de espaço */
#define _P      020     /* Pontuação */
#define _C      040     /* Caractere de controle */
#define _B      0100    /* Branco */
#define _X      0200    /* Digito heXadecimal */
```

A entrada `ctype[]` para cada caracter deve ser a união dos valores da máscara binária que descrevem o caracter. Por exemplo, `'A'` é um caracter maiúsculo (`_U`) bem como um dígito hexadecimal (`_X`), portanto `ctype['A'+1]` deve conter o valor:

```
_U + _X = 01 + 0200 = 0201
```

4.7.5 Suporte à Ordenação de Strings

Se as regras de ordenação para a sua linguagem forem muito complexas para serem tratadas com uma simples tabela `sort_order[]`, será necessário o uso das funções de ordenação de strings.

No momento, a melhor documentação sobre isto são os conjuntos de caracteres que já estão implementados. Confira os conjuntos de caracteres big5, czech, gbk, sjis e tis160 para exemplos.

Você deve especificar o valor `strxfrm_multiply_MYSET=N` no comentário especial no topo do arquivo. `N` deve ser configurado para a razão máxima que as strings podem crescer durante `my_strxfrm_MYSET` (ele deve ser um inteiro positivo).

4.7.6 Suporte à Caracteres Multi-byte

Se você deseja adicionar suporte para novos conjuntos de caracteres que incluem caracteres multi-byte, você precisa usar as funções para caracteres multi-byte.

No momento, a melhor documentação sobre isto são os conjuntos de caracteres que já estão implementados. Confira os conjuntos de caracteres `euc_kr`, `gb2312`, `gbk`, `sjis` e `ujis` para exemplos. Eles são implementados no arquivo `ctype-'conj_caracter'.c` no diretório '`strings`'

Você deve especificar o valor `mbmaxlen_MYSET=N` no comentário especial no topo do arquivo. `N` deve ser configurado como o tamanho em bytes do maior caracter no conjunto.

4.7.7 Problemas com Conjuntos de Caracteres

Se você tentar usar um conjunto de caractere que não foi compilado dentro do se binário, você pode encontrar aluguns problemas:

- Seu programa tem um caminho errado para onde o conjunto de caracter está armazenado (Padrão '`/usr/local/mysql/share/mysql/charsets`'). Isto pode ser corrigido usando a opção `--character-sets-dir` para o programa em questão.
- O conjunto sde caracteres é multi-byte e não pode ser carregado dinamicamente. Neste caso você tem que recompilar o programa com o suporte para o conjunto de caracteres.
- O conjunto de caracteres é dinâmica, mas você não tem um arquivo de configuração para ele. Neste caso você deve instalar o arquivo configure para o conjunto de caracteres de uma nova distriibuição do MySQL.
- Seu arquivo '`Index`' não contém o nome do conjunto de caracteres.

  ```
  ERROR 1105: File '/usr/local/share/mysql/charsets/?.conf' not found
  (Errcode: 2)
  ```

  Neste caso você deve obter um novo arquivo `Index` ou adicionar o nome do conjunto de caracters que falta manualmente.

Para tabelas `MyISAM`, você pode vericifcar o nome e número do conjunto de caracteres para uma tabela com `myisamchk -dvv nome_tabela`.

4.8 Utilitários e Scripts do Lado do Servidor MySQL

4.8.1 Visão Geral dos Scripts e Utilitários do Lado Servidor

Todos os programas MySQL possuem várias opções diferentes, entretanto, todo programa MySQL fornece uma opção `--help` que pode ser usada para obter uma descrição completa das diferentes opções do programa. Por exemplo, experimente `mysql --help`.

Você pode sobrepor ignorar as opções padrões para todos os programas clientes com um arquivo de opções. Seção 4.1.2 [Arquivo de opções], Página 216.

A lista abaixo descreve brevemente os programas MySQL.

`myisamchk`
: Utilitário para descrever, conferir, otimizar e reparar tabelas MySQL. Como o `myisamchk` tem muitas funções, eles são descritos em seu próprio capítulo. Veja Cap“ptexi tulo 4 [Administração do Banco de Dados MySQL], Página 207.

`make_binary_distribution`
: Cria uma edição binária de um MySQL compilado. Isto pode ser enviado por FTP para '`/pub/mysql/Incoming`' em `support.mysql.com` para a conveniência de outros usuários MySQL.

`mysqlbug`
: O script para relatar erros no MySQL. Este script deve sempre ser utilizado quando for necessário preencher um relatório de erros para a lista do MySQL.

`mysqld`
: O servidor (daemon) SQL. Deve sempre estar em execução.

`mysql_install_db`
: Cria as tabelas de permissões do MySQL com os privilégios padrões. Este comando normalmente é executado somente na primeira vez quando o MySQL é instalado em um sistema.

4.8.2 `mysqld-safe`, o wrapper do `mysqld`

`mysqld_safe` é a maneira recomendada para iniciar um daemon `mysqld` no Unix. `mysqld_safe` adiciona alguns recursos de segurança tais como reiniciar o servidor quando um erro ocorrer e log de informações de tempo de execução a um arquivo log.

Note: Antes do MySQL 4.0, `mysqld_safe` é chamado `safe_mysqld`. Para preservar a compatibilidade com versões anteriores, a distribuição binária do MySQL para algumas vezes incluirá `safe_mysqld` como um link simbólico para `mysqld_safe`.

Se você não utilizar `--mysqld=#` ou `--mysql-version=#` o `mysqld_safe` irá utilizar um executável chamado `mysqld-max` se ele existir. Se não, `mysqld_safe` irá iniciar o `mysqld`. Isto torna muito fácil utilizar o `mysql-max` no lugar do `mysqld`; basta copiar `mysqld-max` no mesmo diretório do `mysqld` e ele será utillizado.

Normalmente o script `mysqld_safe` nunca deve ser editado, em vez disto, coloque as opções para o `mysqld_safe` na seção `[mysqld_safe]` no arquivo `my.cnf`. O `mysqld_safe` irá ler todas as opções das seções `[mysqld]`, `[server]` e `[mysqld_safe]` dos arquivos de opções. (Para compatibilidade com versões anteriores, ele também lê as seções `[safe_mysqld]`.) Veja Seção 4.1.2 [Arquivo de opções], Página 216.

Note que todas as opções na linha de comando para o `mysqld_safe` são passadas para o `mysqld`. Se você deseja usar algumas opções no `mysqld_safe` que o `mysqld` não suporte, você deve especificá-las no arquivo de opções.

A maioria das opções para `mysqld_safe` são as mesmas que as do `mysqld`. Veja Seção 4.1.1 [Opções de linha de comando], Página 207.

`mysqld_safe` suporta as seguintes opções:

`--basedir=caminho`
`--core-file-size=#`
: Tamanho do arquivo core que o `mysqld` poderá criar. Passado para `ulimit -c`.

`--datadir=caminho`
`--defaults-extra-file=caminho`
`--defaults-file=caminho`
`--err-log=caminho`
`--log-error=caminho`
: Gava o log de erro no caminho acima. Veja Seção 4.10.1 [Log de erro], Página 372.

`--ledir=caminho`
: Caminho para `mysqld`

`--log=caminho`
`--mysqld=versão_do_mysqld`
: Nome da versão do `mysqld` no diretório `ledir` que você deseja iniciar.

`--mysqld-version=versão`
: Similar ao `--mysqld=` mas aqui você só fornece o sufixo para o `mysqld`. Por exemplo, se você utiliza `--mysqld-version=max`, o `mysqld_safe` irá iniciar a versão `ledir/mysqld-max`. Se o argumento para `--mysqld-version` estiver vazio, `ledir/mysqld` será usado.

`--nice=# (adicionado no MySQL 4.0.14)`
`--no-defaults`
`--open-files-limit=#`
: Número de arquivos que o `mysqld` poderá abrir. Passado para `ulimit -n`. Perceba que será necessário iniciar `mysqld_safe` como root para isto funcionar corretamente!

`--pid-file=caminho`
`--port=#`

`--socket=caminho`
`--timezone=#`
: Configura a variável de fuso horário (`TZ`) para o valor deste parâmetro.

`--user=#`

O script `mysqld_safe` é gravável, portanto ele deve estar apto para iniciar um servidor que foi instalado de uma fonte ou uma versão binária do MySQL, mesmo se o servidor estiver instalado em localizações um pouco diferentes. `mysqld_safe` espera uma destas condições ser verdadeira:

- O servidor e o banco de dados pode ser encontrado relativo ao diretório de onde o `mysqld_safe` foi chamado. `mysqld_safe` procura dentro de seu diretório de trabalho pelos diretórios '`bin`' e '`data`' (para distribuições binárias) ou pelos diretórios '`libexec`'

e '`var`' (para distribuições baseadas em código fonte). Esta condição deve ser satisfeita se você executar o `mysqld_safe` a partir do seu diretório da instalação do MySQL (por exemplo, '`/usr/local/mysql`' para uma distribuição binária).

- Se o servidor e os bancos de dados não forem encontrados relativos ao diretório de trabalho, `mysqld_safe` tenta localizá-lo utilizando caminhos absolutos. Localizações típicas são '`/usr/local/libexec`' e '`/usr/local/var`'. As localizações atuais foram determinadas quando a distribuição foi construída da qual vem o `mysqld_safe`. Eles dever estar corretas se o MySQL foi instalado na localização padrão.

Como o `mysqld_safe` tentará encontrar o servidor e o banco de dados relativo a seu diretório de trabalho, você pode instalar uma distribuição binária do MySQL em qualquer lugar, desde de que o `mysqld_safe` seja iniciado a partir do diretório da instalação:

```
shell> cd diretório_instalação_mysql
shell> bin/mysqld_safe &
```

Se o `mysqld_safe` falhar, mesmo se invocado a partir do diretório de instalação do MySQL, você pode modificá-lo para usar o caminho para o `mysqld` e as opções de caminho que seriam corretas para seu sistema. Perceba que se você atualizar o MySQL no futuro, sua versão modificada de `mysqld_safe` será sobrescrita, portanto, você deve fazer uma cópia de sua versão editada para que você a possa reinstalar.

4.8.3 `mysqld_multi`, programa para gerenciar múltiplos servidores MySQL

`mysqld_multi` gerencia vários processos `mysqld` executando em diferentes sockets UNIX e portas TCP/IP.

O programa irá pesquisar pelos grupos chamados `[mysqld#]` no '`my.cnf`' (ou no arquivo fornecido no parâmetro `--config-file=...`), onde `#` pode ser qualquer número positivo a partir de 1. Este número é referenciado a seguir como número do grupo de opções ou GNR. Números de grupos distinguem grupos de opções para um outro e são usados como argumentos para `mysqld_multi` para especificar quais servidores você deseja iniciar, parar ou obter status. Opções listadas nestes grupos devem ser a mesma que você usaria para iniciar o `mysqld`. (veja Seção 2.4.3 [Automatic start], Página 118). No entanto, para o `mysqld_multi`, esteja certo que cada grupo inclui opções de valores tais como a porta, socket, etc., para ser usado para cada processo `mysqld` individual.

```
Uso: mysqld_multi [OPÇÕES] {start|stop|report} [GNR,GNR,GNR...]
ou   mysqld_multi [OPÇÕES] {start|stop|report} [GNR-GNR,GNR,GNR-GNR,...]
```

O GNR acima significa o número do grupo. Você pode iniciar, parar ou relacionar qualquer GNR ou vários deles ao mesmo tempo. (Veja –example). A lista dos GNR podem ser separadas por vírgulas, ou pelo sinal sinal de menos (-), sendo que o ultimo significa que todos os GNRS entre GNR1-GNR2 serão afetados. Sem o argumento GNR todos os grupos encontrados serão iniciados, parados ou listados. Perceba que você não deve ter nenhum espaço em branco na lista GNR. Qualquer coisa depois de um espaço em branco é ignorado.

`mysqld_multi` suporta as seguintes opções:

`--config-file=...`
Arquivo de configuração alternativo. NOTA: Isto não irá afetar as próprias opções do programa (grupo `[mysqld_multi]`), mas somente grupos [mysqld#]. Sem esta opção tudo será procurado a partir do arquivo my.cnf.

`--example`
Fornece um exemplo de um arquivo de configuração.

`--help` Exibe esta ajuda e sai.

`--log=...`
Arquivo Log. Deve ser informado o caminho completo e o nome do arquivo log. NOTA: se o arquivo existir, tudo será anexado.

`--mysqladmin=...`
Binário `mysqladmin` a ser usado para o desligamento do servidor.

`--mysqld=...`
Binário `mysqld` a ser usado. Lembre-se que você também pode fornecer `mysqld_safe` a esta opção. As opções são passadas ao `mysqld`. Apenas tenha certeza que o `mysqld` está localizado na sua variável de ambiente `PATH` ou corrija o `mysqld_safe`.

`--no-log` Imprime na saída padrão em vez do arquivo log. Por padrão o arquivo log sempre fica ligado.

`--password=...`
Senha do usuário para o `mysqladmin`.

`--tcp-ip` Conecta ao(s) servidor(es) MySQL através de porta TCP/IP no lugar de socket UNIX. Isto afeta a ação de desligar e relatar. Se um arquivo socket estiver faltando, o servidor pode ainda estar executando, mas só pode ser acessado através da porta TCP/IP. Por padrão a conexão é feita através de socket UNIX.

`--user=...`
Usuário MySQL para o `mysqladmin`.

`--version`
Exibe o número da versão e sai.

Algumas notas sobre `mysqld_multi`:

- Tenha certeza que o usuário MySQL, que finalizar os serviços `mysqld` (e.g. utilizando o `mysqladmin`) tem a mesma senha e usuário para todos os diretórios de dados acessados (para o banco de dados 'mysql'). E tenha certeza que o usuário tem o privilégio 'Shutdown_priv'! Se você possui diversos diretórios de dados e vários bancos de dados 'mysql' com diferentes senhas para o usuário 'root' do MySQL, você pode desejar criar um usuário comum 'multi-admin' para cada um que utilize a mesma senha (veja abaixo). Exemplo de como fazer isto:

```
shell> mysql -u root -S /tmp/mysql.sock -psenha_root -e
"GRANT SHUTDOWN ON *.* TO multi_admin@localhost IDENTIFIED BY 'multipass'"
```

Veja Seção 4.3.6 [Privilégios], Página 232.

Você deve fazer isto para cada servidor `mysqld` executando em cada diretório de dados, que você tem (Apenas altere o socket, -S=...)

- `pid-file` é muito importante, se você estiver utilizando `mysqld_safe` para iniciar o `mysqld` (ex. –mysqld=mysqld_safe) Todos os `mysqld` devem ter seus próprios `pid-file`. A vantagem de utilizar o `mysqld_safe` no lugar de executar diretamente o `mysqld` é que `mysqld_safe` guarda todos os processos e irá reiniciá-los, se um processo do `mysqld` falhar devido a um sinal kill -9, ou similar. (Como um falha de segmentação, que nunca pode acontecer com o MySQL.) Por favor note que pode ser necessário executar o script `mysqld_safe` de um lugar específico. Isto significa que você pode ter que alterar o diretório atual para um diretório específico antes de iniciar o `mysqld_multi`. Se você tiver problemas ao iniciar, por favor veja o script `mysqld_safe`. Verifique especialmente as linhas:

```
------------------------------------------------------------------------
MY_PWD=`pwd` Check if we are starting this relative (for the binary
release) if test -d /data/mysql -a -f ./share/mysql/english/errmsg.sys
-a -x ./bin/mysqld
------------------------------------------------------------------------
```

  Veja Seção 4.8.2 [`mysqld_safe`], Página 331. O teste acima deve ser bem sucedido, ou você pode encontrar problemas.
- Esteja certo do perigoso de iniciar múltiplos `mysqlds` no mesmo diretório de dados. Utilize diretórios de dados diferentes, a menos que você realmente **SAIBA** o que está fazendo!
- O arquivo de socket e a porta TCP/IP devem ser diferentes para cada `mysqld`.
- O primeiro e quinto grupo `mysqld` foram intencionalmente deixados de lado no exemplo. Você pode ter lacunas no arquivo de configuração. Isto lhe permite mais flexibilidade. A ordem na qual os `mysqlds` são iniciados ou desligados depende da ordem em que eles aparecem no arquivo de configuração.
- Quando você desejar referenciar a um grupo específico utilizando GNR com este programa, basta utilizar o número no fim do nome do grupo ([mysqld# <==).
- Você pode desejar utilizar a opção '–user' para o `mysqld`, mas para isto você precisa ser o usuário root quando iniciar o script `mysqld_multi`. Não importa se a opção existe no arquivo de configuração; você receberá apenas um alerta se você não for o superusuário e o `mysqlds` for iniciado com a **SUA** conta no Unix. **IMPORTANTE**: Tenha certeza que o `pid-file` e o diretório de dados é acessível para leitura e escrita (+execução para o diretório) para **ESTE** usuário UNIX que iniciará o processo `mysqld`. **NÃO** utilize a conta de root para isto, a menos que você **SAIBA** o que está fazendo!
- **MAIS IMPORTANTE**: Tenha certeza que você entendeu os significados das opções que são passadas para os `mysqlds` e porque **VOCÊ PRECISARIA** ter processos `mysqld` separados. Iniciando múltiplos `mysqlds` em um diretório de dados **NÃO IRÁ** melhorar a performance em um sistema baseado em threads.

Veja Seção 4.2 [Servidores múltiplos], Página 219.

Este é um exemplo do arquivo de configuração para o funcionamento do `mysqld_multi`.

```
# Este arquivo provavelmente deve estar em seu diretório home (~/.my.cnf) ou /etc/m
# Version 2.1 by Jani Tolonen
```

```
[mysqld_multi]
mysqld     = /usr/local/bin/mysqld_safe
mysqladmin = /usr/local/bin/mysqladmin
user       = multi_admin
password   = multipass

[mysqld2]
socket     = /tmp/mysql.sock2
port       = 3307
pid-file   = /usr/local/mysql/var2/hostname.pid2
datadir    = /usr/local/mysql/var2
language   = /usr/local/share/mysql/english
user       = john

[mysqld3]
socket     = /tmp/mysql.sock3
port       = 3308
pid-file   = /usr/local/mysql/var3/hostname.pid3
datadir    = /usr/local/mysql/var3
language   = /usr/local/share/mysql/swedish
user       = monty

[mysqld4]
socket     = /tmp/mysql.sock4
port       = 3309
pid-file   = /usr/local/mysql/var4/hostname.pid4
datadir    = /usr/local/mysql/var4
language   = /usr/local/share/mysql/estonia
user       = tonu

[mysqld6]
socket     = /tmp/mysql.sock6
port       = 3311
pid-file   = /usr/local/mysql/var6/hostname.pid6
datadir    = /usr/local/mysql/var6
language   = /usr/local/share/mysql/japanese
user       = jani
```

Veja Seção 4.1.2 [Arquivos de opções], Página 216.

4.8.4 `myisampack`, O Gerador de Tabelas Compactadas de Somente Leitura do MySQL

`myisampack` é usado para compactar tabelas MyISAM, e `pack_isam` é usado para compactar tabelas ISAM. Como as tabelas ISAM estão ultrapassadas, nós iremos discutir aqui somente

sobre o `myisampack`, mas tudo dito sobre `myisampack` também pode ser verdadeiro para o `pack_isam`.

`myisampack` trabalha compactando cada coluna na tabela separadamente. A informação necessária para descompactar colunas é lida em memória quando a tabela é aberta. Isto resulta em uma performance muito melhor quando estiver acessando registros individuais, porque você precisará descompactar somente um registro, não um bloco muito maior do disco como faz o Stacker no MS-DOS. Normalmente, `myisampack` compacta o arquivo de dados 40%-70%.

O MySQL utiliza mapeamento de memória (`nmap()`) em tabelas compactadas e retorna ao uso normal de leitura e escrita se `nmap()` não funcionar.

Por favor, note o seguinte:

- Depois de comapctada, a tabela é somente-leitura. Isto é, normalmente, pretendido (como quando acessamos tabelas compactadas em um CD). Permitir que se faça gravação em uma tabela compactada também está em nossa lista TODO, mas com baixa prioridade.
- `myisampack` também pode compactar colunas `BLOB` ou `TEXT`. O antigo `pack_isam` (para tabelas `ISAM`) não pode fazer isto.

`myisampack` é chamado desta forma:

```
shell> myisampack [opções] nome_arquivo ...
```

Cada nome_arquivo deve ter o nome de um arquivo de índice (‘`.MYI`’). Se você não se encontra em um diretório de bancos de dados, você deve especificar o caminho completo para o arquivo. Pode-se omitir a extensão ‘`.MYI`’.

`myisampack` suporta as seguintes opções:

`-b, --backup`
: Realiza um backup da tabela como `nome_tabela.OLD`.

`-#, --debug=debug_options`
: Log da saída de depuração. A string `debug_options` geralmante é `'d:t:o,nome_arquivo'`.

`-f, --force`
: Força a compactação da tabela mesmo se ela se tornar maior ou se o arquivo temporário existir. `myisampack` cria um arquivo temporário chamado ‘`nome_tabela.TMD`’ enquanto ele compacta a tabela. Se você matar o `myisampack` o arquivo ‘`.TMD`’ não pode ser removido. Normalmente, `myisampack` sai com um erro se ele descobrir que ‘`nome_tabela.TMD`’ existe. Com `--force`, `myisampack` compacta a tabela de qualquer maneira.

`-?, --help`
: Exibe uma mensagem de ajuda e sai.

`-j nome_tabela_grande, --join=nome_tabela_grande`
: Une todas as tabelas nomeadas na linha de comando em uma única tabela `nome_tabela_grande`. Todas tabelas que forem combinadas DEVEM ser idênticas (mesmos nomes de colunas e tipos, alguns índices, etc.).

`-p #, --packlength=#`
: Especifica o comprimento do tamanho de armazenamento, em bytes. O valor deve ser 1, 2 ou 3. (`myisampack` armazena todas as linhas com ponteiros de tamanhos 1, 2 ou 3 bytes. Na maioria dos casos normais, `myisampack` pode determinar o valor correto do tamanho antes de começar a compactar o arquivo, mas ele pode notificar durante o processo de compactação que ele pode ter usado um tamanho menor. Neste caso `myisampack` irá exibir uma nota dizendo que a próxima vez que você compactar o mesmo arquivo você pode utilizar um registro de tamanho menor.)

`-s, --silent`
: Modo silencioso. Escreve a saída somente quando algum erro ocorrer.

`-t, --test`
: Não compacta realmente a tabela, apenas testa a sua compactação.

`-T dir_name, --tmp_dir=dir_name`
: Utiliza o diretório especificado como a localização em que serão gravadas as tabelas temporárias.

`-v, --verbose`
: Modo verbose. Escreve informação sobre o progresso e resultado da compactação.

`-V, --version`
: Exibe informação de versão e sai.

`-w, --wait`
: Espera e tenta novamente se a tabela estiver em uso. Se o servidor `mysqld` foi iniciado com a opção `--skip-locking`, não é uma boa idéia chamar `myisampack` se a tabela puder ser atualizada durante o processo de compactação.

A seqüência de comandos mostrados abaixo ilustra uma típica seção de compactação de tabelas:

```
shell> ls -l station.*
-rw-rw-r--   1 monty    my         994128 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          53248 Apr 17 19:00 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

shell> myisamchk -dvv station

MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-02-02  3:06:43
Data records:              1192  Deleted blocks:              0
Datafile: Parts:           1192  Deleted data:                0
Datafile pointer (bytes):     2  Keyfile pointer (bytes):     2
Max datafile length:   54657023  Max keyfile length:   33554431
Recordlength:               834
Record format: Fixed length
```

```
table description:
Key Start Len Index   Type                          Root  Blocksize    Rec/key
1   2     4   unique  unsigned long                 1024       1024          1
2   32    30  multip. text                         10240       1024          1

Field Start Length Type
1     1     1
2     2     4
3     6     4
4     10    1
5     11    20
6     31    1
7     32    30
8     62    35
9     97    35
10    132   35
11    167   4
12    171   16
13    187   35
14    222   4
15    226   16
16    242   20
17    262   20
18    282   20
19    302   30
20    332   4
21    336   4
22    340   1
23    341   8
24    349   8
25    357   8
26    365   2
27    367   2
28    369   4
29    373   4
30    377   1
31    378   2
32    380   8
33    388   4
34    392   4
35    396   4
36    400   4
37    404   1
38    405   4
39    409   4
40    413   4
```

```
41    417   4
42    421   4
43    425   4
44    429   20
45    449   30
46    479   1
47    480   1
48    481   79
49    560   79
50    639   79
51    718   79
52    797   8
53    805   1
54    806   1
55    807   20
56    827   4
57    831   4

shell> myisampack station.MYI
Compressing station.MYI: (1192 records)
- Calculating statistics

normal:     20  empty-space:      16  empty-zero:        12  empty-fill:  11
pre-space:   0  end-space:        12  table-lookups:      5  zero:         7
Original trees:  57  After join: 17
- Compressing file
87.14%

shell> ls -l station.*
-rw-rw-r--   1 monty    my         127874 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          55296 Apr 17 19:04 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

shell> myisamchk -dvv station

MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-04-17 19:04:26
Data records:              1192  Deleted blocks:              0
Datafile: Parts:           1192  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     1
Max datafile length:   16777215  Max keyfile length:     131071
Recordlength:               834
Record format: Compressed

table description:
```

```
Key Start Len Index   Type                          Root  Blocksize    Rec/key
1   2     4   unique  unsigned long                 10240       1024          1
2   32    30  multip. text                          54272       1024          1

Field Start Length Type                              Huff tree  Bits
1     1     1      constant                                  1     0
2     2     4      zerofill(1)                               2     9
3     6     4      no zeros, zerofill(1)                     2     9
4     10    1                                                3     9
5     11    20     table-lookup                              4     0
6     31    1                                                3     9
7     32    30     no endspace, not_always                   5     9
8     62    35     no endspace, not_always, no empty         6     9
9     97    35     no empty                                  7     9
10    132   35     no endspace, not_always, no empty         6     9
11    167   4      zerofill(1)                               2     9
12    171   16     no endspace, not_always, no empty         5     9
13    187   35     no endspace, not_always, no empty         6     9
14    222   4      zerofill(1)                               2     9
15    226   16     no endspace, not_always, no empty         5     9
16    242   20     no endspace, not_always                   8     9
17    262   20     no endspace, no empty                     8     9
18    282   20     no endspace, no empty                     5     9
19    302   30     no endspace, no empty                     6     9
20    332   4      always zero                               2     9
21    336   4      always zero                               2     9
22    340   1                                                3     9
23    341   8      table-lookup                              9     0
24    349   8      table-lookup                             10     0
25    357   8      always zero                               2     9
26    365   2                                                2     9
27    367   2      no zeros, zerofill(1)                     2     9
28    369   4      no zeros, zerofill(1)                     2     9
29    373   4      table-lookup                             11     0
30    377   1                                                3     9
31    378   2      no zeros, zerofill(1)                     2     9
32    380   8      no zeros                                  2     9
33    388   4      always zero                               2     9
34    392   4      table-lookup                             12     0
35    396   4      no zeros, zerofill(1)                    13     9
36    400   4      no zeros, zerofill(1)                     2     9
37    404   1                                                2     9
38    405   4      no zeros                                  2     9
39    409   4      always zero                               2     9
40    413   4      no zeros                                  2     9
41    417   4      always zero                               2     9
42    421   4      no zeros                                  2     9
```

```
43    425   4      always zero                    2     9
44    429   20     no empty                       3     9
45    449   30     no empty                       3     9
46    479   1                                    14     4
47    480   1                                    14     4
48    481   79     no endspace, no empty         15     9
49    560   79     no empty                       2     9
50    639   79     no empty                       2     9
51    718   79     no endspace                   16     9
52    797   8      no empty                       2     9
53    805   1                                    17     1
54    806   1                                    3     9
55    807   20     no empty                       3     9
56    827   4      no zeros, zerofill(2)          2     9
57    831   4      no zeros, zerofill(1)          2     9
```

A informação exibida pelo `myisampack` é descrita abaixo:

`normal` O número de colunas para qual nenhum empacotamento extra é utilizado.

`empty-space`
O número de colunas contendo valores que são somente espaços; estes ocuparão apenas 1 bit.

`empty-zero`
O número de colunas contendo valores que são somente 0's binários; ocuparão 1 bit.

`empty-fill`
O número de colunas inteiras que não ocupam a faixa completa de bytes de seu tipo; estes são alteradas para um tipo menor (por exemplo, uma coluna `INTEGER` pode ser alterada para `MEDIUMINT`).

`pre-space`
O número de colunas decimais que são armazenadas com espaços a esquerda. Neste caso, cada valor irá conter uma contagem para o número de espaços.

`end-space`
O número de colunas que tem muitos espaços espaços extras. Neste caso, cada valor conterá uma contagem para o número de espaços sobrando.

`table-lookup`
A coluna tem somente um pequeno número de valores diferentes, que são convertidos para um `ENUM` antes da compressão Huffman.

`zero` O número de colunas em que todos os valores estão zerados.

`Original trees`
O número inicial de árvores Huffman.

`After join`
O número de árvores Huffman distintas que sobram depois de unir árvores para poupar espaço de cabeçalho.

Depois que uma tabela foi compactada, `myisamchk -dvv` exibe informações adicionais sobre cada campo:

`Type` O tipo de campo deve conter as seguites descrições:

`constant` Todas linhas tem o mesmo valor.

`no endspace`
Não armazena espaços no fim.

`no endspace, not_always`
Não armazena espaços no fim e não faz compactação de espaços finais para todos os valores.

`no endspace, no empty`
Não armazena espaços no fim. Não armazena valores vazios.

`table-lookup`
A coluna foi convertida para um `ENUM`.

`zerofill(n)`
Os `n` bytes mais significativos no valor são sempre 0 e não são armazenados.

`no zeros` Não armazena zeros.

`always zero`
Valores zero são armazenados em 1 bit.

`Huff tree` A árvore Huffman associada com o campo.

`Bits` O número de bits usado na árvore Huffman.

Depois de ter executado `pack_isam`/`myisampack` você deve executar o `isamchk`/`myisamchk` para recriar o índice. Neste momento você pode também ordenar os blocos de índices para criar estatísticas necessárias para o otimizador do MySQL trabalhar de maneira mais eficiente.

```
myisamchk -rq --analyze --sort-index nome_tabela.MYI
isamchk   -rq --analyze --sort-index nome_tabela.ISM
```

Depois de instalar a tabela compactada no diretório de banco de dados MySQL você deve fazer `mysqladmin flush-tables` para forçar o `mysqld` a iniciar usando a nova tabela.

Se você desejar descompactar uma tabela compactada, você pode fazer isto com a opção `--unpack` para o `isamchk` ou `myisamchk`.

4.8.5 `mysqld-max`, om servidor `mysqld` extendido

`mysqld-max` é o servidor MySQL (`mysqld`) configurado com as seguintes opções de configuração:

Opção	**Comentário**
–with-server-suffix=-max	Adiciona um sufixo à string de versão `mysqld`
–with-innodb	Suporte a tabelas InnoDB
–with-bdb	Suporte para tabelas Berkeley DB (BDB)

CFLAGS=-DUSE_SYMDIR	Suporte a links simbólicos para Windows

A opção para habilitar o suporte ao InnoDB é necessário apenas no MySQL 3.23. No MySQL 4 e acima, o InnoDB já é incluído por padrão.

Você pode encontrar os binários do MySQL-max em `http://www.mysql.com/downloads/mysql-max-4.0.h`

A distribuição binária Windows MySQL 3.23 inclui tanto o binário `mysqld.exe` padrão e o binário `mysqld-max.exe`. `http://www.mysql.com/downloads/mysql-4.0.html`. Veja Seção 2.1.1 [Instalação no Windows], Página 60.

Note que como o Berkeley DB (BDB) não está disponível para todas plataformas, alguns dos binários `Max` podem não ter suporte para ela. Você pode conferir quais tipos de tabelas são suportadas executando a seguinte consulta:

```
mysql> SHOW VARIABLES LIKE "have_%";
+------------------+----------+
| Variable_name    | Value    |
+------------------+----------+
| have_bdb         | NO       |
| have_crypt       | YES      |
| have_innodb      | YES      |
| have_isam        | YES      |
| have_raid        | NO       |
| have_symlink     | DISABLED |
| have_openssl     | NO       |
| have_query_cache | YES      |
+------------------+----------+
```

O significado dos valores na segunda coluna são:

Valor	**Significado**.
`YES`	A opção está ativa e é utilizada.
`NO`	O MySQL não está compilado com suporte a esta opção.
`DISABLED`	A opção xxx está desabilitada porque o `mysqld` foi iniciado com `--skip-xxxx` ou porque não foi iniciado com todas as opções necessárias para habilitar esta opção. Neste caso o arquivo `hostname.err` deve conter uma razão indicando o porque da opção estar desabilitada.

NOTA: Para conseguir criar tabelas InnoDB você **DEVE** editar suas opções de inicialização para incluir ao menos a opção `innodb_data_file_path`. Veja Seção 7.5.2 [InnoDB no MySQL 3.23], Página 643.

Para obter melhor performance para tabelas BDB, você deve adicionar algumas opções de configuração para elas também .Veja Seção 7.6.3 [Iniciando BDB], Página 699.

`mysqld_safe` tenta iniciar automaticamente qualquer binário `mysqld` com o prefixo `-max`. Isto faz com que seja fácil testar um outro binário `mysqld` em uma instalação existente. Apenas execute o `configure` com as opções deseejadas e, então, instale o novo binário `mysqld` como `mysqld-max` no mesmo diretório onde seu antigo binário `mysqld` está. Veja Seção 4.8.2 [`mysqld_safe`], Página 331.

No Linux, o RPM `mysqld-max` utiliza o recurso `mysqld_safe` já mencionado. (Ele apenas instala o executável `mysqld-max` e o `mysqld_safe` usará automaticamente este executável quando o `mysqld_safe` for reiniciado).

A tabela a seguir mostra quais tipos de tabelas nossos binários **MySQL-Max** incluem:

Sistema	`BDB`	`InnoDB`
Windows/NT	S	S
AIX 4.3	N	S
HP-UX 11.0	N	S
Linux-Alpha	N	S
Linux-Intel	S	S
Linux-IA-64	N	S
Solaris-Intel	N	S
Solaris-SPARC	S	S
SCO OSR5	S	S
UnixWare	S	S
Mac OS X	N	S

Note que a partir do MySQL 4, você não precisa de um servidos MySQL Max para o InnoDB porque ele é incluído por padrão.

4.9 Utilitários e Scripts do Lado do Cliente MySQL

4.9.1 Visão Geral dos Utilitários e Scripts do Lado do Cliente

Todos clientes MySQL que comunicam com o servidor utilizando a biblioteca `mysqlclient` utilizam as seguintes variáveis de ambiente:

Nome	**Descrição**
`MYSQL_UNIX_PORT`	O socket padrão, utilizado para conexões ao `localhost`
`MYSQL_TCP_PORT`	A porta TCP/IP padrão
`MYSQL_PWD`	A senha padrão
`MYSQL_DEBUG`	Opções de depuração-ratreamento durante depuração
`TMPDIR`	O diretório onde tabelas e arquivos temporários são criados

A utilização de `MYSQL_PWD` é insegura. Veja Seção 4.3.8 [Connecting], Página 238.

No Unix, o cliente '`mysql`' utiliza o arquivo nomeado na variável de ambiente `MYSQL_HISTFILE` para salvar o histórico da linha de comando. O valor padrão para o arquivo de histórico é '`$HOME/.mysql_history`', onde `$HOME` é o valor da variável de ambiente `HOME`. Veja Apêndice F [Environment variables], Página 1092.

Se você não quiser manter um arquivo que contenh um registro de suas consultas, primeiro remova '`.mysql_history`' se ele existir, então use uma das seguintes técnicas:

- Defina a variável `MYSQL_HISTFILE` para '`/dev/null`'. Para que esta configuração tenha efeito a cada vez que você logar, coloque-a em um dos arquivos de inicialização da sua shell.
- Crie '`.mysql_histfile`' como um link simbólico para '`/dev/null`':

  ```
  shell> ln -s /dev/null $HOME/.mysql_history
  ```

  Você só precisa de fazer isto uma vez.

Todos os programas MySQL podem receber várias opções diferentes. Entretanto, todo programa MySQL fornece a opção `--help` que você pode utilizar para obter uma descrição completa das diferentes opções do programa. Por exemplo, experimente `mysql --help`

Você pode sobrepor todas as opções padrões para programas cliente padrões com um arquivo de opções. Seção 4.1.2 [Option files], Página 216

A lista abaixo descreve resumidamente os programas MySQL:

`msql2mysql`
: Um script shell que converte programas `mSQL` para MySQL. Ele não lida com todos os casos, mas ele fornece um bom inicio para a conversão.

`mysql`
: A ferramenta de linha de comando para a entrada de consultas interativamente ou a execução de consultas a partir de um arquivo no modo batch. Veja Seção 4.9.2 [`mysql`], Página 347.

`mysqlcc`
: Este programa fornece uma interface gráfica para interagir com o servidor. server. Veja Seção 4.9.3 [`mysqlcc`], Página 354.

`mysqlaccess`
: Um script que verifica os privilégios de acesso para uma combinação de nome de máquina, usuário e banco de dados.

`mysqladmin`
: Utilitário para realizar operações administrativas, tais como criação ou remoção de bancos de dados, recarga das tabelas de permissões, descarga de tabelas em disco e reabertura dos arquivos log. `mysqladmin` também pode ser usado para exibir informações de versão, processos e estado do servidor. Veja Seção 4.9.4 [`mysqladmin`], Página 356.

`mysqlbinlog`
: Utilitário para leitura das consultas de um log binário. Pode ser usado para recuperação de falhas com um backup antigo. Veja Seção 4.9.5 [`mysqlbinlog`], Página 358.

`mysqldump`
: Descarrega um banco de dados MySQL em um arquivo como instruções SQL ou como arquivo texto separado por tabulação. Versão aprimorada do freeware escrito originalmente por Igor Romanenko. Veja Seção 4.9.7 [`mysqldump`], Página 361.

`mysqlimport`
: Importa arquivos texto em suas tabelas respectivas utilizando `LOAD DATA INFILE`. Veja Seção 4.9.9 [`mysqlimport`], Página 367.

`mysqlshow`
: Exibe informações sobre bancos de dados, tabelas, colunas e índices.

`replace`
: Um programa utilitário que é usado pelo `msql2mysql`, mas que também pode ser aplicável mais genericamente. `replace` altera conjuntos de caracteres. Utiliza uma máquina de estado finito para comparar strings maiores primeiro. Pode ser usada para trocar conjuntos de caracteres. Por exemplo, este comando troca `a` e `b` nos arquivos dados:

```
shell> replace a b b a -- arquivo1 arquivo2 ...
```

4.9.2 `mysql`, A Ferramenta de Linha de Comando

O `mysql` é uma shell SQL simples (com capacidades GNU `readline`). Ele suporta usos interativos e não interativos. Quando usado interativamente, os resultados das consultas são apresentadas no formato de tabela ASCII. Quando não usado interativamente (como um filtro por exemplo), o resultado é apresentado em um formato separado por tabulações. (O formato de saída pode ser alterado utilizando opções da linha de comando.) Você pode executar scripts desta forma:

```
shell> mysql database < script.sql > saida.tab
```

Se você tiver problemas devido a memória insuficiente no cliente, utilize a opção `--quick`! Isto força o `mysql` a utilizar `mysql_use_result()` no lugar de `mysql_store_result()` para recuperar o conjunto de resultados.

Utilizar o `mysql` é muito fáci. Inicie-o como mostrado a seguir: `mysql banco_de_dados` ou `mysql --user=nome_usuário --password=sua_senha banco_de_dados`. Digite uma instrução SQL, termine-a com ';', '\g', ou '\G' e pressione RETURN/ENTER.

O `mysql` Suporta as seguintes opções:

`-?, --help`
: Exibe esta ajuda e sai.

`-A, --no-auto-rehash`
: Sem reprocessamento automático. O 'rehash' deve ser usado se o usuário desejar que o cliente `mysql` complete as tabelas e campos. Esta opção é usada para acelerar a inicialização do cliente.

`--prompt=...`
: Configura o prompt do mysql com o formato especificado.

`-b, --no-beep`
: Deliga o beep nos erros.

`-B, --batch`
: Exibe resultados com o caractere de tabulação como o separador, cada registro em uma nova linha. Não utiliza o arquivo de histórico.

`--character-sets-dir=...`
: Diretório onde os conjuntos de caracteres estão localizados.

`-C, --compress`
: Utiliza compactação no protocolo cliente/servidor.

`-#, --debug[=...]`
: Log de Depuração. O padrão é 'd:t:o,/tmp/mysql.trace'.

`-D, --database=...`
: Qual banco de dados usar. Isto geralmente é util em um arquivo `my.cnf`.

`--default-character-set=...`
: Configura o conjunto de caracters padrão.

`-e, --execute=...`
: Executa o comando e sai. (Saída parecida com –batch)

`-E, --vertical`
: Exibe a saída de uma consulta (linhas) verticalmente. Sem esta opção você também pode forçar esta saída terminando suas instruções com `\G`.

`-f, --force`
: Continue mesmo se for obtido um erro SQL.

`-g, --no-named-commands`
: Comandos nomeados serão desabilitados. Utilize somente a forma *, ou use comandos nomeados apenas no começo da linha terminada com um ponto-e-vírgula (;). Desde a versão 10.9, o cliente agora inicia com esta opção habilitada por padrão! Com a opção -g, entretando, comandos de formato longo continuarão funcionando na primeira linha.

`-G, --enable-named-commands`
: Comandos nomeados são **habilitados**. Comandos de formato longo são aceitos assim como os comandos reduzidos *.

`-i, --ignore-space`
: Ignore caractere de espaço depois de nomes de funções.

`-h, --host=...`
: Conectar à máquina especificada.

`-H, --html`
: Produz saída HTML.

`-X, --xml` Produz saída XML.

`-L, --skip-line-numbers`
: Não escreve o número da linha para os erros. Útil quando se deseja comparar arquivos com resultados que incluem mensagens de erro.

`--no-pager`
: Desabilita paginação e impressão na saída padrão. Veja também a ajuda interativa (\h).

`--no-tee` Desabilita arquivo de saída. Veja também a ajuda interativa (\h).

`-n, --unbuffered`
: Descarrega e atualiza o buffer depois de cada pesquisa.

`-N, --skip-column-names`
: Não escrever nomes de colunas nos resultados.

`-O, --set-variable nome=opção`
: Fornece um valor a uma variável. `--help` lista as variáveis. Por favor, note que as sintaxes `--set-variable=name=value` e `-O name=value` estão obsoletas desde o MySQL 4.0, use `--nome=valor`.

`-o, --one-database`
: Atualiza somente o banco de dados padrão. Isto é útil para evitar atualização em outros bancos de dados no log de atualizações.

`--pager[=...]`
: Tipo de saída. O padrão é sua variável de ambiente `PAGER`. Paginadores válidos são: less, more, cat [>nome_arquivo], etc. Veja também a ajuda interativa (\h). Esta opção não funciona no modo batch. A opção pager funciona somente no UNIX.

`-p[password], --password[=...]`
: Senha a ser usada ao conectar ao servidor. Se uma senha não é fornecida na linha de comando, lhe será solicitado uma. Perceba que se você utilizar o formato curto `-p` você não pode ter um espaço entre a opção e a senha.

`-P --port=...`
: Número da porta TCP/IP para usar na conexão.

`--protocol=(TCP | SOCKET | PIPE | MEMORY)`
: Especifica o protocolo de conexão usado. Novo no MySQL 4.1.

`-q, --quick`
: Não faz cache do resultado, imprime linha a linha. Isto pode deixar o servidor mais lento se a saída for suspendida. Não usa arquivo de histórico.

`-r, --raw` Exibe valores de colunas sem conversão de escapes. Utilizado com `--batch`

`--reconnect`
: Se a conexão é perdida, tentar reconectar ao servidor automaticamente (mas apenas uma vez).

`-s, --silent`
: Opção para ser mais silencioso.

`-S --socket=...`
: Arquivo socket para ser utilizado na conexão.

`-t --table`
: Saída no formato de tabela. Isto é padrão no modo não-batch.

`-T, --debug-info`
: Exibe alguma informação de depuração na saída.

`--tee=...`
: Anexa tudo no arquivo de saída. Veja também a ajuda interativa (\h). Não funciona no modo batch.

`-u, --user=#`
: Usuário para login diferente do usuário atual do sistema.

`-U, --safe-updates[=#], --i-am-a-dummy[=#]`
: Permite somente que `UPDATE` e `DELETE` utilizem chaves. Veja abaixo para maiores informações sobre esta opção. Você pode zerar esta opção se possui-la no arquivo `my.cnf` utilizando `--safe-updates=0`.

`-v, --verbose`
: Modo verbose (-v -v -v fornece o formato de saída da tabela).

`-V, --version`
: Gera saída com informação de versão e sai.

`-w, --wait`

Espera e repete em vez de sair se a conexão estiver inacessível.

Você também pode configurar as seguntes variáveis com `-O` ou `--set-variable`. Por favor, note que as sintaxes `--set-variable=nome=valor` e `-O name=value` estão obsoletas desde o MySQL 4.0, use `--var=option`:

Nome Variável	Padrão	Descrição
connect_timeout	0	Número de seguntos antes de esgotar o tempo da conexão
local-infile	0	Disabilita (0) ou habilita (1) capacidade `LOCAL` para `LOAD DATA INFILE`
max_allowed_packet	16777216	Tamanho máximo do pacote para enviar/receber do servidor
net_buffer_length	16384	Tamanho do buffer para comunicação TCP/IP e socket
select_limit	1000	Limite automático para `SELECT` quando utilizar `--safe-updtaes`
max_join_size	1000000	Limite automático para registros em uma join quando utilizar `--safe-updtaes`.

Se o cliente `mysql` perder a conexào com o servidor enquanto envia uma consulta, ele tentará se reconectar imediatamente e automaticamente uma vez e enviar a consulta novamente. Note que mesmo se ele obter sucesso na reconexão, como sua primeira conexão foi finalizada, todas seus objetos da sessão anteriores foram perdidos: tabelas temporárias, e variáveis de sessão e de usuário. Desta forma, o comportamento acima pode ser perigoso para você, como neste exemplo onde o servidor foi desligado e reiniciado sem você saber:

```
mysql> set @a=1;
Query OK, 0 rows affected (0.05 sec)

mysql> insert into t values(@a);
ERROR 2006: MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    1
Current database: test

Query OK, 1 row affected (1.30 sec)

mysql> select * from t;
+------+
| a    |
+------+
| NULL |
+------+
1 row in set (0.05 sec)
```

A variável de usuário `@a` foi perdida com a conexão e depois da reconexão ela é indefinida. Para se proteger deste risco, você pode iniciar o cliente `mysql` com a opção `--disable-reconnect`.

Se você digitar 'help' na linha de comando, `mysql` irá exibir os comandos que ele suporta:

```
mysql> help
```

```
MySQL commands:
help      (\h)    Display this text.
?         (\h)    Synonym for 'help'.
clear     (\c)    Clear command.
connect   (\r)    Reconnect to the server.
                  Optional arguments are db and host.
delimiter (\d)    Set query delimiter.
edit      (\e)    Edit command with $EDITOR.
ego       (\G)    Send command to mysql server,
          display result vertically.
exit      (\q)    Exit mysql. Same as quit.
go        (\g)    Send command to mysql server.
nopager   (\n)    Disable pager, print to stdout.
notee     (\t)    Don't write into outfile.
pager     (\P)    Set PAGER [to_pager].
          Print the query results via PAGER.
print     (\p)    Print current command.
prompt    (\R)    Change your mysql prompt.
quit      (\q)    Quit mysql.
rehash    (\#)    Rebuild completion hash.
source    (\.)    Execute an SQL script file.
  Takes a file name as an argument.
status    (\s)    Get status information from the server.
system    (\!)    Execute a system shell command.
tee       (\T)    Set outfile [to_outfile].
  Append everything into given outfile.
use       (\u)    Use another database.
       Takes database name as argument.
```

Os comandos `edit`, `nopager`, `pager`, e `system` funcionam apenas no Unix.

O comando `status` lhe fornece algumas informações sobre a conexão e o servidor que está utilizando. Se você estiver executando no modo `--safe-updates`, `status` irá também imprimir os valores para as variáveis `mysql` que afetam suas consultas.

Uma opção útil para iniciantes (introduzido no MySQL versão 3.23.11) é o `--safe-updates` (ou `--i-am-a-dummy` para usuários que uma vez possam ter feito um `DELETE FROM nome_tabela` mas esqueceram da cláusula `WHERE`). Quando utilizar esta opção, o `mysql` envia o seguinte comando ao servidor MySQL quando abrir a conexão.

```
SET SQL_SAFE_UPDATES=1,SQL_SELECT_LIMIT=#select_limit#,
    SQL_MAX_JOIN_SIZE=#max_join_size#"
```

onde `#select_limit#` e `#max_join size#` são variáveis que podem ser configuradas da linha de comando `mysql`. Veja Seção 5.5.6 [SET OPTION], Página 460.

O efeito da opção acima é:

- Você não tem permissão de utilizar uma instrução `UPDATE` ou `DELETE` se você não possuir uma chave na parte `WHERE`. Pode-se, entretanto, forçar um `UPDATE/DELETE` utilizando `LIMIT`:

```
UPDATE nome_tabela SET campo_nao_chave=# WHERE campo_nao_chave=# LIMIT 1;
```

- Todos resultados maiores são limitados automaticamente a `#select_limit#` linhas.
- `SELECT`'s que provavelmente precisarão examinar mais que `#max_join_size` combinaçoes de linhas serão abortadas.

Algumas dicas úteis sobre o cliente `mysql`:

Alguns dados são muito mais legíveis quando exibido verticalmente, em vez da saída do tipo caixa horizontal comum. Por exemplo: Textos longos, que incluem várias linhas, são muito mais fáceis de serem lidos com saída vertical.

```
mysql> SELECT * FROM mails WHERE LENGTH(txt) < 300 lIMIT 300,1\G
*************************** 1. row ***************************
  msg_nro: 3068
     date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Monty
    reply: monty@no.spam.com
  mail_to: "Thimble Smith" <tim@no.spam.com>
      sbj: UTF-8
      txt: >>>>> "Thimble" == Thimble Smith writes:

Thimble> Hi.  I think this is a good idea.  Is anyone familiar with UTF-8
Thimble> or Unicode? Otherwise, I'll put this on my TODO list and see what
Thimble> happens.

Yes, please do that.

Regards,
Monty
     file: inbox-jani-1
     hash: 190402944
1 row in set (0.09 sec)
```

Para o log, você pode utilizar a opção `tee`. O `tee` pode ser iniciado com a opção `--tee=...`, ou pela linha de comando de maneira interativa com o comando `tee`. Todos os dados exibidos na tela serão anexados no arquivo fornecido. Isto também pode ser muito útil para propósitos de depuração. O `tee` pode ser desabilitado da linha de comando com o comando `notee`. Executando `tee` novamente o log é reiniciado. Sem um parâmetro o arquivo anterior será usado. Perceba que `tee` irá atualizar os resultados dentro do arquivo depois de cada comando, pouco antes da linha de comando reaparecer esperando pelo próximo comando.

Navegar ou pesquisar os resultados no modo interativo em algum programa do UNIX como o less, more ou outro similar, é agora possível com a opção `--pager[=...]`. Sem argumento, o cliente `mysql` irá procurar pela variável de ambiente `PAGER` e configurar `pager` para este valor. `pager` pode ser iniciado a partir da linha de comando interativa com o comando `pager` e desabilitado com o comando `nopager`. O comando recebe um argumento opcional e e o `pager` será configurado com ele. O comando `pager` pode ser chamado com um argumento, mas isto requer que a opção `--pager` seja usada, ou o `pager` será usado com a saída padrão. `pager` funciona somente no UNIX, uma vez que é utilizado a função `popen()`,

que não existe no Windows. No Windows a opção `tee` pode ser utilizada, entretanto ela pode não ser cômoda como `pager` pode ser em algumas situações.

Algumas dicas sobre `pager`:

- Você pode usá-lo para gravar em um arquivo:

  ```
  mysql> pager cat > /tmp/log.txt
  ```

  e os resultados irão somente para um arquivo. Você também pode passar qualquer opções para os programas que você deseja utilizar com `pager`:

  ```
  mysql> pager less -n -i -S
  ```

- Note a opção `-S` exibida acima. Você pode achá-la muito útil quando navegar pelos resultados; experimente com a opção com saída a horizontal (finalize os comandos com `\g`, ou `;`) e com saída vertical (final dos comandos com `\G`). Algumas vezes um resultado com um conjunto muito largo é difícil ser lido na tela, com a opção `-S` para `less`, você pode navegar nos resultados com o `less` interativo da esquerda para a direita, evitando que linhas maiores que sua tela continuem na próxima linha. Isto pode tornar o conjunto do resultado muito mais legível. você pode alterar o modo entre ligado e desligado com o `less` interativo com `-S`. Veja o 'h'(help) para mais ajuda sobre o `less`.

  Você pode combinar maneiras muito complexas para lidar com os resultados, por exemplo, o seguinte enviaria os resultados para dois arquivos em dois diferentes diretórios, em dois discos diferentes montados em /dr1 e /dr2, e ainda exibe o resultado na tela via less:

  ```
  mysql> pager cat | tee /dr1/tmp/res.txt | \
  tee /dr2/tmp/res2.txt | less -n -i -S
  ```

Você também pode combinar as duas funções acima; tenha o `tee` habilitado, o `pager` configurado para 'less' e você estará apto a navegar nos resultados no less do Unix e ainda ter tudo anexado em um arquivo ao mesmo tempo. A diferença entre `UNIX tee` usado com o `pager` e o `tee` embutido no cliente `mysql` é que o `tee` embutido funciona mesmo se você não tiver o comando `UNIX tee` disponível. O `tee` embutido também loga tudo que é exibido na tela, e o `UNIX tee` usado com `pager` não loga completamente. Por último o `tee` interativo é mais cômodo para trocar entre os modos on e off, quando você desejar logar alguma coisa em um arquivo, mas deseja estar apto para desligar o recurso quando necessário.

A partir da versão 4.0.2 é possível alterar o prompt no cliente de linha de comando `mysql`.

Você pode usar as seguintes opções do prompt:

Opção	**Descrição**
\v	versão mysqld
\d	banco de dados em uso
\h	máquina na qual está conectado
\p	porta na qual está conectado
\u	nome do usuário
\U	nome_usuário@maquina
\\	'\'
\n	nova quebra de linha
\t	tab
\	espaço

_	espaço
\R	hora no formato 24h (0-23)
\r	hora no formato 12h (1-12)
\m	minutos
\y	ano com dois digitos
\Y	ano com quatro digitos
\D	formato completo da data
\s	segundos
\w	dia da semana no formato com 3 letras (Mon, Tue, ...)
\P	am/pm
\o	mês no formato de número
\O	mês no formato com 3 letras (Jan, Feb, ...)
\c	contador que cresce a cada comando

'\' seguido por qualquer outra letra apenas retorna aquela letra.

Você pode definir o prompt nos seguintes lugares:

Variável de Ambiente

Você pode configurar o `prompt` em qualquer arquivo de configuração do MySQL, no grupo `mysql`. Por exemplo:

```
[mysql]
prompt=(\u@\h) [\d]>\_
```

Linha de Comando

Você pode definir a opção `--prompt` na linha de comando para `mysql`. Por exemplo:

```
shell> mysql --prompt="(\u@\h) [\d]> "

(usuário@maquina) [banco de dados]>
```

Interativamente

Você também pode usar o comando `prompt` (ou `\R`) para alterar o seu prompt interativamente. Por exemplo:

```
mysql> prompt (\u@\h) [\d]>\_
PROMPT set to '(\u@\h) [\d]>\_'
(usuario@maquina) [banco de dados]>
(usuario@maquina) [banco de dados]> prompt
Returning to default PROMPT of mysql>
mysql>
```

4.9.3 `mysqlcc`, The MySQL Control Center

`mysqlcc`, o Centro de Controle do MySQL, é um cliente independente de plataforma que fornece um interface gráfica ao usuário (GUI) para o servidor de banco de dados MySQL. Ela suporta uso interativo, incluindo destaque de sintaxe e complementação com tab. Ele fornece gerenciamento de banco de dados e tabelas e permite a administração do servidor.

Atualmente, o `mysqlcc` executa em plataformas Windows e Linux.

`mysqlcc` não está incluído com a distribuição MySQL, mas pode ser feito o download separadamente em `http://www.mysql.com/downloads/`.

`mysqlcc` suporta as seguintes opções:

`-?, --help`
: Exibe esta ajuda e sai.

`-b, --blocking_queries`
: Usa consultas em bloco.

`-C, --compress`
: Usa o protocolo servidor/cliente compactado.

`-c, --connection_name=name`
: Este é um sinônimo para `--server`.

`-d, --database=...`
: Banco de dados a ser usado. Isto é útil principalmente no arquivo '`my.cnf`'.

`-H, --history_size=#`
: Tamanho do histórico para a janiela de consultas.

`-h, --host=...`
: Conecta a uma determinda máquina.

`-p[password], --password[=...]`
: Senha usada ao se conectar ao servidor. Se uma senha não for especificada na linha de comando, você deverá informá-la. Note que se você usar a forma simplificada `-p` não é permitido um espaço entre a opçõa e a senha.

`-g, --plugins_path=name`
: Caminho para o diretório onde os plugins do MySQL Control Center estao lacalizados.

`-P port_num, --port=port_num`
: Número da porta TCP/IP para uso na conexão.

`-q, --query`
: Abre uma janela de consulta na inicialização.

`-r, --register`
: Abre a caixa de diálogo 'Register Server' na inicialização.

`-s, --server=name`
: Nome da conexão do MySQL Control Center.

`-S --socket=...`
: Arquivo socket usado na conexão.

`-y, --syntax`
: Habilita destque da sintaxe e complementação

`-Y, --syntax_file=name`
: Arquivo de sintaxe para complementação.

`-T, --translations_path=name`
Caminho para o diretório onde as traduções do MySQL Control Center estão localizados.

`-u, --user=#`
Usuário para login se diferente do usuário atual.

`-V, --version`
Exibe a versão e sai.

Você também pode configurar as seguntes variáveis com `-O` ou `--set-variable`. Por favor, note que as sintaxes `--set-variable=nome=valor` e `-O name=value` estão obsoletas desde o MySQL 4.0, use `--var=option`:

Variable Name	Default	Description
connect_timeout	0	Number of seconds before connection timeout.
local-infile	0	Disable (0) or enable (1) `LOCAL` capability for `LOAD DATA INFILE`
max_allowed_packet	16777216	Max packet length to send to/receive from server
net_buffer_length	16384	Buffer for TCP/IP and socket communication
select_limit	1000	Automatic limit for `SELECT` when using `--safe-updtaes`
max_join_size	1000000	Automatic limit for rows in a join when using `--safe-updates`

4.9.4 `mysqladmin`, Administrando um Servidor MySQL

Um utilitário para realizar operações administrativas. A sintaxe é:

```
shell> mysqladmin [OPÇÕES] comando [opção_do_comando] comando...
```

Você pode obter uma lista das opção que sua versão do `mysqladmin` suporta executando `mysqladmin --help`.

O `mysqladmin` atual suporta os seguintes comandos:

`create databasename`
Cria um novo banco de dados.

`drop databasename`
Apaga um banco de dados e todas suas tabelas.

`extended-status`
Fornece uma mensagem extendida sobre o estado do servidor.

`flush-hosts`
Atualiza todos os nomes de máquinas que estiverem no cache.

`flush-logs`
Atualiza todos os logs.

`flush-tables`
Atualiza todas as tabelas.

`flush-privileges`
Recarrega tabelas de permissões (mesmo que reload).

`kill id,id,...`
: Mata threads do MySQL.

`password` Configura uma nova senha. Altera a antiga senha para nova senha.

`ping` Checa se o mysqld está ativo.

`processlist`
: Exibe lista de threads ativas no servidor, com a instrução `SHOW PROCESSLIST`. Se a opção `--verbose` é passada, a saída é como aquela de `SHOW FULL PROCESSLIST`.

`reload` Recarrega tabelas de permissão.

`refresh` Atualiza todas as tabelas e fecha e abre arquivos de log.

`shutdown` Desliga o servidor.

`slave-start`
: Inicia thread de replicação no slave.

`slave-stop`
: Termina a thread de replicação no slave.

`status` Fornece uma mensagem curta sobre o estado do servidor.

`variables`
: Exibe variáveis disponíveis.

`version` Obtêm informação de versão do servidor.

Todos comandos podem ser reduzidos para seu prefixo único. Por exemplo:

```
shell> mysqladmin proc stat
+----+-------+-----------+----+-------------+------+-------+------+
| Id | User  | Host      | db | Command     | Time | State | Info |
+----+-------+-----------+----+-------------+------+-------+------+
| 6  | monty | localhost |    | Processlist | 0    |       |      |
+----+-------+-----------+----+-------------+------+-------+------+
Uptime: 10077  Threads: 1  Questions: 9  Slow queries: 0
Opens: 6 Flush tables: 1  Open tables: 2
Memory in use: 1092K  Max memory used: 1116K
```

O resultado do comando `mysqladmin status` possui as seguintes colunas:

Uptime	Número de segundos que o servidor MySQL está funcionando.
Threads	Número de threads ativas (clientes).
Questions	Número de solicitações dos clientes desde que o `mysqld` foi iniciado.
Slow queries	Consultas que demoram mais que `long_query_time` segundos. Veja Seção 4.10.5 [Log de consultas lentas], Página 377.
Opens	Quantas tabelas foram abertas pelo `mysqld`.
Flush tables	Número de comandos `flush...`, `refresh` e `reload`.
Open tables	Número de tabelas abertas atualmente.

Memory in use	Memória alocada diretamente pelo código do `mysqld` (disponível somente quando o MySQL é compilado com –with-debug=full).
Max memory used	Memória máxima alocada diretamente pelo código do `mysqld` (disponível somente quando o MySQL é compilado com –with-debug=full).

Se você executa um `mysqladmin shutdown` em um socket (em outras palavras, em um computador onde o `mysqld` está executando), `mysqladmin` irá esperar até que o `arquivo-pid` do MySQL seja removido para garantir que o servidor `mysqld` parou corretamente.

4.9.5 `mysqlbinlog`, Executando as Consultas a Partir de um Log Binário

Você pode examinad o arquivo de log binário (veja Seção 4.10.4 [Binary log], Página 374) com o utilitário `mysqlbinlog`.

```
shell> mysqlbinlog hostname-bin.001
```

exibirá todas as consultas contidas no log binário '`hostname-bin.001`', junto com outras informações (tempo da consulta, ID da thread que a executou, o timestamp de quando foi executada, etc).

Você pode colocar a saída do `mysqlbinlog` em um cliente `mysql`; isto é usado para recuperações de falhas quando você tem um backup antigo (veja Seção 4.5.1 [Backup], Página 275):

```
shell> mysqlbinlog hostname-bin.001 | mysql
```

ou

```
shell> mysqlbinlog hostname-bin.[0-9]* | mysql
```

Você também pode redirecionar a saída do `mysqlbinlog` para um arquivo texto, então modifique este arquivo texto (para excluir as consultas que você não quer executar por alguma razão), e então execute as consultas a partir do arquivo texto dentro do `mysql`.

`mysqlbinlog` possui a opção `position=#` que exibirá apenas as consultas cujo offset no log binário é maior ou igual a `#`.

Se você tiver mais que um log binário para executar no servidor MySQL, o método seguro é fazê-lo em uma única conexão MySQL. Aqui está o que pode ser INseguro:

```
shell> mysqlbinlog hostname-bin.001 | mysql # DANGER!!
shell> mysqlbinlog hostname-bin.002 | mysql # DANGER!!
```

Isto causará problemas se o primeiro log binário conter um `CREATE TEMPORARY TABLE` e o segundo contém uma consulta que utiliza esta tabela temporária: quando o primeiro `mysql` termina, ele apara a tabela temporária, assim a o segundo `mysql` relatará um "tabela desconhecida". Isto ocorre porque você deve executar todos os log binários que você deseja em uma única conexão, especialmente se você usa tabelas temporárias. Aqui estão dois modos possíveis:

```
shell> mysqlbinlog hostname-bin.001 hostname-bin.002 | mysql
```

```
shell> mysqlbinlog hostname-bin.001 >  /tmp/queries.sql
shell> mysqlbinlog hostname-bin.002 >> /tmp/queries.sql
shell> mysql -e "source /tmp/queries.sql"
```

A partir do MySQL 4.0.14, `mysqlbinlog` pode preparar uma entrada para o `mysql` executar um `LOAD DATA INFILE` a partir de um log binário. Como o log binário contém os dados para carregar (isto é verdade para o MySQL 4.0; o MySQL 3.23 não grava o dado carregado em um log binário, assim o arquivo original era necessário quando se queria executar o conteúdo do log binário), `mysqlbinlog` copiará este data para um arquivo temporário e imprime um comando `LOAD DATA INFILE` para o `mysql` carregar este arquivo temporário. O local onde o arquivo temorário é criado é o diretório temporário por padrão; ele pode ser alterado com a opção `local-load` do `mysqlbinlog`.

Antes do MySQL 4.1, `mysqlbinlog` não podia preaparar saída cabíveis para `mysql` quando o log binário continha consultas de diferentes threads usando tabelas temporárias de mesmo nome, se estas consultas eram entrelaçadas. Isto está resolvido no MySQL 4.1.

Você também pode usar o `mysqlbinlog --read-from-remote-server` para ler o log binário diretamente de um servidor MySQL remoto. No entanto, isto é algo que está obsoleto já que queremos tornar fácil de se aplicar os logs binários em servidores MySQL em execução.

`mysqlbinlog --help` lhe dará mais informações

4.9.6 Usando `mysqlcheck` para Manutenção de Tabelas e Recuperação em Caso de Falhas

Desde o MySQL versão 3.23.38 você estará apto a usar a nova ferramenta de reparos e verificação de tabelas `MyISAM`. A diferença para o `myisamchk` é que o `mysqlcheck` deve ser usado quando o servidor `mysqld` estiver em funcionamento, enquanto o `myisamchk` deve ser usado quando ele não estiver. O benefício é que você não precisará mais desligar o servidor `mysqld` para verificar ou reparar suas tabelas.

O `mysqlcheck` utiliza os comandos do servidor MySQL `CHECK`, `REPAIR`, `ANALYZE` e `OPTIMIZE` de um modo conveniente para o usuário.

Existem três modos alternativos de chamar o `mysqlcheck`:

```
shell> mysqlcheck [OPÇÕES] database [tabelas]
shell> mysqlcheck [OPÇÕES] --databases DB1 [DB2 DB3...]
shell> mysqlcheck [OPÇÕES] --all-databases
```

Pode ser usado de uma maneira muito similar ao `mysqldump` quando o assunto for quais bancos de dados e tabelas devem ser escolhidas.

O `mysqlcheck` tem um recurso especial comparado comparado aos outros clientes; o comportamento padrão, verificando as tabelas (-c), pode ser alterado renomeando o binário. Se você deseja ter uma ferramenta que repare as tabelas como o procedimento padrão, você deve copiar o `mysqlcheck` para o disco com um outro nome, `mysqlrepair`, ou crie um link simbólico com o nome `mysqlrepair`. Se você chamar `mysqlrepair` agora, ele irá reparar as tabelas como seu procedimento padrão.

Os nomes que podem ser utilizados para alterar o comportamento padrão do `mysqlcheck` são:

```
mysqlrepair:   A opção padrão será -r
mysqlanalyze:  A opção padrão será -a
mysqloptimize: A opção padrão será -o
```

As opções disponíveis para o `mysqlcheck` estão listadas aqui, por favor verifique o que a sua versão suporta com o `mysqlcheck --help`.

`-A, --all-databases`
: Verifica todos os bancos de dados. Isto é o mesmo que –databases com todos os bancos de dados selecionados.

`-1, --all-in-1`
: Em vez de fazer uma consulta para cada tabela, execute todas as consultas separadamente para cada banco de dados. Nomes de tabelas estarão em uma lista separada por vírgula.

`-a, --analyze`
: Análise as tabelas fornecidas.

`--auto-repair`
: Se uma tabela checada está corrompida, ela é corrigida automaticamente. O reparo será feito depois que todas as tabelas tiverem sido checadas e forem detectadas tabelas corrompidas.

`-#, --debug=...`
: Log de saída de depuração. Normalmente é 'd:t:o,filename'

`--character-sets-dir=...`
: Diretório onde estão os conjuntos de caracteres.

`-c, --check`
: Verifca erros em tabelas

`-C, --check-only-changed`
: Verifica somente tabelas que foram alteradas desde a última conferência ou que não foram fechada corretamente.

`--compress`
: Utilize compressão no protocolo server/cliente.

`-?, --help`
: Exibe esta mensagem de ajuda e sai.

`-B, --databases`
: Para verificar diversos bancos de dados. Perceba a diferença no uso; Neste caso nenhuma tabela será fornecida. Todos os argumentos são tratados como nomes de bancos de dados.

`--default-character-set=...`
: Configura o conjunto de caracteres padrão.

`-F, --fast`
: Verifica somente as tabelas que não foram fechadas corretamente

`-f, --force`
: Continue mesmo se nós obtermos um erro de sql.

`-e, --extended`
: Se você estiver utilizando esta opção com CHECK TABLE, irá garantir que a tabela está 100 por cento consistente, mas leva bastante tempo.

  Se você utilizar esta opção com REPAIR TABLE, ele irá executar um comando de reparos na tabela, que não só irá demorar muito tempo para executar, mas também pode produzir muitas linhas de lixo.

`-h, --host=...`
Conecta à máquina.

`-m, --medium-check`
Mais rápido que verificação extendida, mas encontra somente 99.99 de todos os erros. Deve resolver a maioria dos casos.

`-o, --optimize`
Otimizador de tabelas

`-p, --password[=...]`
Senha para usar ao conectar ao servidor. Se a senha não for fornecida será solicitada no terminal.

`-P, --port=...`
Número de porta para usar para conexão.

`-q, --quick`
Se esta opção for utilizada com CHECK TABLE, evita a busca de registros verificando links errados. Esta é a conferência mais rápida.

Se você estiver utilizando esta opção com REPAIR TABLE, ela tentará reparar somente a árvore de índices. Este é o método de reparo mais rápido para uma tabela.

`-r, --repair`
Pode corrigir quase tudo exceto chaves únicas que não são únicas.

`-s, --silent`
Exibe somente mensagens de erro.

`-S, --socket=...`
Arquivo socket para usar na conexão.

`--tables` Sobrepõe a opção –databases (-B).

`-u, --user=#`
Usuário para o login, se não for o usuário atual.

`-v, --verbose`
Exibe informação sobre os vários estágios.

`-V, --version`
Exibe informação sobre a versão e sai.

4.9.7 `mysqldump`, Descarregando a Estrutura de Tabelas e Dados

Utilitário para descarregar um banco de dados ou uma coleção de bancos de dados para backup ou transferencia para outro servidor SQL (Não necessariamente um servidor MySQL). A descarga irá conter instruções SQL para cria a tabela e/ou popular a tabela.

Se a idéia é backup do servidor, deve ser considerada a utilização do `mysqlhotcopy`. Veja Seção 4.9.8 [`mysqlhotcopy`], Página 366.

```
shell> mysqldump [OPÇÕES] banco_de_dados [tabelas]
OR     mysqldump [OPÇÕES] --databases [OPÇÕES] BD1 [BD2 BD3...]
OR     mysqldump [OPÇÕES] --all-databases [OPÇÕES]
```

Se você não fornecer nenhuma tabela ou utilizar o `--databases` ou `--all-databases`, todo(s) o(s) banco(s) de dados será(ão) descarregado(s).

Você pode obter uma lista das opções que sua versão do `mysqldump` suporta executando `mysqldump --help`.

Perceba que se você executar o `mysqldump` sem a opção `--quick` ou `--opt`, o `mysqldump` irá carregar todo o conjunto do resultado na memória antes de descarregar o resultado. Isto provavelmente será um problema se você está descarregando um banco de dados grande.

Note que se você estiver utilizando uma cópia nova do programa `mysqldump` e se você for fazer uma descarga que será lida em um servidor MySQL muito antigo, você não deve utilizar as opções `--opt` ou `-e`.

`mysqldump` suporta as seguintes opções:

`--add-locks`
: Adicione `LOCK TABLES` antes de `UNLOCK TABLE` depois de cada descarga de tabelas. (Para obter inserções mais rápidas no MySQL.)

`--add-drop-table`
: Adicione um `drop table` antes de cada instrução create.

`-A, --all-databases`
: Descarrega todos os bancos de dados. Isto irá ser o mesmo que `--databases` com todos os bancos de dados selecionados.

`-a, --all` Inclui todas as opções do create específicas do MySQL.

`--allow-keywords`
: Permite criação de nomes que colunas que são palavras chaves. Isto funciona utilizando o nome da tabela como prefixo em cada nome de coluna.

`-c, --complete-insert`
: Utilize instruções de insert completas (com nomes de colunas).

`-C, --compress`
: Compacta todas as informações entre o cliente e o servidor se ambos suportarem a compactação.

`-B, --databases`
: Para descarregar diversos bancos de dados. Perceba a diferença no uso. Neste caso nenhuma tabela é fornecida. Todos argumentos são estimados como nomes de bancos de dados. `USE nome_bd;` será incluído na saída antes de cada banco de dados novo.

`--delayed`
: Insere registros com o comando `INSERT DELAYED`.

`-e, --extended-insert`
: Utiliza a nova sintaxe multilinhas `INSERT`. (Fornece instruções de inserção mais compactas e mais rápidas.)

`-#, --debug[=option_string]`
: Rastreia a utilização do programa (para depuração).

`--help`
: Exibe uma mensagem de ajuda e sai.

`--fields-terminated-by=...`
`--fields-enclosed-by=...`
`--fields-optionally-enclosed-by=...`
`--fields-escaped-by=...`
`--lines-terminated-by=...`
: Estas opções são usadas com a opção `-T` e tem o mesmo significado que as cláusulas correspondentes em `LOAD DATA INFILE` Veja Seção 6.4.8 [`LOAD DATA`], Página 588.

`-F, --flush-logs`
: Atualiza o arquivo de log no servidor MySQL antes de iniciar a descarga.

`-f, --force,`
: Continue mesmo se obter um erro de SQL durantes uma descarga de tabela.

`-h, --host=..`
: Descarrega dados do servidor MySQL na máquina especificada. A máquina padrão é `localhost`.

`-l, --lock-tables.`
: Bloqueia todas as tabelas antes de iniciar a descarga. As tabelas são bloqueadas com `READ LOCAL` para permitir inserções concorrentes no caso de tabelas `MyISAM`.

  Por favor, note que ao descarregar multiplas tabelas, `--lock-tables` bloqueará as tabelas de cada banco de dados separadamente. Assim, usar esta opção não garantirá que suas tabelas sejam logicamente consistentes entre os banco de dados. Tabela me diferentes bancos de dados podem ser descarregadas em estados completamente diferentes.

`-K, --disable-keys`
: `/*!40000 ALTER TABLE nome_tb DISABLE KEYS */;` e `/*!40000 ALTER TABLE nome_tb ENABLE KEYS */;` será colocado na saída. Isto fará com que a carga de dados no MySQL 4.0 server seja mais rápida já que os índices são criados depois que todos os dados são inseridos.

`-n, --no-create-db`
: `'CREATE DATABASE /*!32312 IF NOT EXISTS*/ nome_bd;'` não será colocado na saída. A linha acima será adicionada se a opção `--databases` ou `--all-databases` for fornecida.

`-t, --no-create-info`
: Não grava informações de criação de tabelas (A instrução `CREATE TABLE`.)

`-d, --no-data`
: Não grava nenhuma informação de registros para a tabela. Isto é muito útil se você desejar apenas um dump da estrutura da tabela!

`--opt`
: O mesmo que `--quick --add-drop-table --add-locks --extended-insert --lock-tables`. Fornece a descarga mais rápida para leitura em um servidor MySQL.

`-pyour_pass, --password[=sua_senha]`
: A senha para usar quando conectando ao servidor. Se não for especificado a parte '`=sua_senha`', o `mysqldump` irá perguntar por uma senha.

`-P port_num, --port=porta_num`
: O número da porta TCP/IP usado para conectar a uma máquina. (Isto é usado para conexões a máquinas diferentes de `localhost`, na qual sockets Unix são utilizados.)

`-q, --quick`
: Não utiliza buffers para as consultas, descarrega diretamente para saída padrão. Utilize `mysql_use_result()` para fazer isto.

`-Q, --quote-names`
: Coloca os nomes de colunas e tabelas entre ‘‘.

`-r, --result-file=...`
: Direcione a saída para um determinado arquivo. Esta opção deve ser usada no MSDOS porque previne a conversão de nova linha '\n' para '\n\r' (nova linha + retorno de carro).

`--single-transaction`
: Esta opção envia um comando SQL `BEGIN` antes de carregar os dados do servidor. Ele é mais útil com tabelas `InnoDB` e nível `READ_COMMITTED` de isolação da transação, já que neste modo ela fará um dump do estado de consistência do banco de dados no momento que o `BEGIN` for enviado sem bloquear qualquer aplicação.

  Ao usar esta opção você deve manter em mente que será feito um dump no estado consistente apenas das tabelas transacionais, ex., qualquer tabela `MyISAM` ou `HEAP` na qual for feito um dump durante está p[ção pode ainda mudar de estado.

  A opção `--single-transaction` foi adicionada na versão 4.0.2. Esta opção é mutualmente exclusiva com a opção `--lock-tables` já que `LOCK TABLES` já faz um commit da transação anterior internamente.

`-S /path/to/socket, --socket=/path/to/socket`
: O arquivo socket que será utilizado quando conectar à `localhost` (que é a máquina padrão).

`--tables` Sobrepõe a opção –databases (-B).

`-T, --tab=path-to-some-directory`
: Cria um arquivo `nome_tabela.sql`, que contém os comandos SQL CREATE e um arquivo `nome_tabela.txt`, que contém os dados, para cada tabela dada. O formato do arquivo `.txt` é feito de acordo com as opções `--fields-xxx` e `--lines--xxx`. **Nota**: Esta opção só funciona se `mysqldump` está sendo executado na mesma máquina que o daemon `mysqld`. Você deve usar uma conta MySQL que tem o privilégio `FILE`, e o login de usuário/grupo com o qual o `mysqld` está sendo executado (normalmente usuário `mysql`, grupo `mysql`) precisa ter permissão para criar/gravar um arquivo no local especificado.

`-u user_name, --user=user_name`
: O nome do usuário do MySQL para usar ao conectar ao servidor. O valor padrão é seu nome de usuário no Unix.

`-O nome=valor, --set-variable=nome=valor`
: Confirgura o valor de uma variável. As variáveis possíveis são listadas abaixo. Note que a sintaxe `--set-variable=nome=valor` e `-O nome=valor` está obsoleto desde o MySQL 4.0. Use `--nome=valor`.

`-v, --verbose`
: Modo verbose. Exibe mais informações sobre o que o programa realiza.

`-V, --version`
: Exibe informações de versão e sai.

`-w, --where='where-condition'`
: Faz um dump apenas dos registros selecionados. Note que as aspas são obrigatórias:

```
"--where=user='jimf'" "-wuserid>1" "-wuserid<1"
```

`-X, --xml` Faz um dump do banco de dados no formato XML

`-x, --first-slave`
: Faz um lock de todas as tabelas de todos os bancos de dados.

`--master-data`
: Como `--first-slave`, mas também exibe algum comando `CHANGE MASTER TO` o qual, mais tarde, fará o seu slave iniciar a partir da posição certa no log binário do master, se você tiver configurado o seu slave usando este dump SQL do master.

`-O net_buffer_length=#, where # < 16M`
: Quando estiver criando instruções de inserções em múltiplas linhas (com a opção `--extended-insert` ou `--opt`), `mysqldump` irá criar linhas até o tamanho de `net_buffer_length`. Se você aumentar esta variável, você também deve se assegurar que a variável `max_allowed_packet` no servidor MySQL é maior que a `net_buffer_length`.

O uso mais comum do `mysqldump` é provavelmente para fazer backups de bancos de dados inteiros. Veja Seção 4.5.1 [Backup], Página 275.

```
mysqldump --opt banco_dados > arquivo-backup.sql
```

Você pode ler de volta no MySQL com:

```
mysql banco_dados < arquivo-backup.sql
```

ou

```
mysql -e "source /path-to-backup/backup-file.sql" database
```

Entretanto, é muito útil também popular outro servidor MySQL com informações de um banco de dados:

```
mysqldump --opt banco_dados | mysql ---host=máquina-remota -C banco_dados
```

É possível descarregar vários bancos de dados com um comando:

```
mysqldump --databases banco_dados1 [banco_dados2 banco_dados3...] > meus_bancosdeda
```

Se desejar descarregar todos os bancos de dados, pode-se utilizar:

```
mysqldump --all-databases > todos_bancos_dados.sql
```

4.9.8 `mysqlhotcopy`, Copiando Bancos de Dados e Tabelas do MySQL

O `mysqlhotcopy` é um script perl que utiliza `LOCK TABLES`, `FLUSH TABLES` e `cp` ou `scp` para fazer um backup rápido de um banco de dados. É a maneira mais rápida para fazer um backup do banco de dados e de algumas tabelas mas ele só pode ser executado na mesma máquina onde os diretórios dos bancos de dados estão. O `mysqlhotcopy` só funciona no Unix e apenas para as tabelas `MyISAM` e `ISAM`.

```
mysqlhotcopy nome_bd [/caminho/para/novo_diretório]

mysqlhotcopy nome_bd_2 ... nome_bd_2 /caminho/para/novo_diretório

mysqlhotcopy nome_bd./regex/
```

`mysqlhotcopy` suporta as seguintes opções:

`-?, --help`
: Exibe uma tela de ajuda e sai

`-u, --user=#`
: Usuário para fazer login no banco de dados

`-p, --password=#`
: Senha para usar ao conectar ao servidor

`-P, --port=#`
: Porta para usar ao conectar ao servidor local

`-S, --socket=#`
: Qual socket usar ao conectando a um servidor local

`--allowold`
: Não aborta se o alvo já existir (renomeie-o para _old)

`--keepold`
: Não apaga alvos anteriores (agora renomeados) quando pronto

`--noindices`
: Não inclui arquivos de índices na cópia para deixar o backup menor e mais rápido. Os índices podem ser recostruídos mais tarde com `myisamchk -rq`..

`--method=#`
: Metódo para copiar (`cp` ou `scp`).

`-q, --quiet`
: Seja silencioso exceto em erros

`--debug` Habilita depuração

`-n, --dryrun`
: Relata ações sem realizá-las

`--regexp=#`
: Copia todos bancos de dados com nomes que coincidem com a expressão regular

`--suffix=#`
: Sufixo para nomes de bancos de dados copiados

`--checkpoint=#`
: Insere entrada de ponto de controle um uma bd.tabela especificada

`--flushlog`
: Atualiza logs uma vez que todas as tabelas estiverem bloqueadas.

`--tmpdir=#`
: Diretório Temporário (em vez de /tmp).

Você pode utilizar `perldoc mysqlhotcopy` para obter uma documentação mais completa de `mysqlhotcopy`.

`mysqlhotcopy` lê os grupos `[client]` e `[mysqlhotcopy]` dos arquivos de opções.

Para poder executar `mysqlhotcopy` é necessário acesso de escrita ao diretório de backup, privilégio `SELECT` nas tabelas que desejar copiar e o privilégio `Reload` no MySQL (para poder executar `FLUSH TABLES`).

4.9.9 `mysqlimport`, Importando Dados de Arquivos Texto

`mysqlimport` fornece uma interface de linha de comando para a instrução SQL `LOAD DATA INFILE`. A maioria das opções aceitas correspondem diretamente às opções de `LOAD DATA INFILE`. Veja Seção 6.4.8 [`LOAD DATA`], Página 588.

`mysqlimport` é chamado desta maneira:

```
shell> mysqlimport [opções] banco_de_dados arquivo_texto1 [arquivo_texto2....]
```

Para cada arquivo texto passadoo na linha de comando, `mysqlimport` remove qualquer extensão do nome do arquivo e utiliza o resultado para determinar para qual tabela os dados do arquivo serão importados. Por exemplo, arquivos chamados '`patient.txt`', '`patient.text`' e '`patient`' serão importados para uma tabela chamada `patient`.

`mysqlimport` suporta as seguintes opções:

`-c, --columns=...`
: Esta opção recebe uma lista de nomes de campos separados por vírgula como um argumento. A lista de campos é utilizada para criar um comando `LOAD DATA INFILE` adequado que é então passado ao MySQL. Veja Seção 6.4.8 [`LOAD DATA`], Página 588.

`-C, --compress`
: Compacta todas as informações entre o cliente e o servidor se ambos suportarem compressão.

`-#, --debug[=option_string]`
: Rastreia o programa (para depuração).

`-d, --delete`
: Esvazie a tabela antes de importar o arquivo texto.

`--fields-terminated-by=...`
`--fields-enclosed-by=...`
`--fields-optionally-enclosed-by=...`
`--fields-escaped-by=...`
`--lines-terminated-by=...`
: Estas opções tem o mesmo significado que as cláusulas correspondentes para `LOAD DATA INFILE`. Veja Seção 6.4.8 [`LOAD DATA`], Página 588.

`-f, --force`
: Ignorar erros. Por exemplo, se uma tabela para um arquivo texto não existir, continue processando quaisquer arquivos restantes. Sem `--force`, `mysqlimport` sai se uma tabela não existir.

`--help`
: Exibe uma mensagem de ajuda e sai.

`-h host_name, --host=host_name`
: Importa dados para o servidor MySQL na máquina referida. A máquina padrão é `localhost`.

`-i, --ignore`
: Veja a descrição para a opção `--replace`.

`--ignore-lines=n`
: Ignora as primeiras `n` linhas do arquivo de dados.

`-l, --lock-tables`
: Bloqueia **TODAS** as tabelas para escrita antes de processar qualquer arquivo texto. Isto garante que todas as tabelas são sincronizadas no servidor.

`-L, --local`
: Lê arquivos de entrada do cliente. Por padrão, é assumido que os arquivos texto estão no servidor se você conectar à `localhost` (máquina padrão).

`-pyour_pass, --password[=sua_senha]`
: Senha para conectar ao servidor. Se você não especificar a parte '`=sua_senha`', o `mysqlimport` irá pedir por uma senha.

`-P port_num, --port=port_num`
: O número da porta TCP/IP para usar quando conectar a uma máquina.

`--protocol=(TCP | SOCKET | PIPE | MEMORY)`
: Para especificar o protocolo de conexão. Novo no MySQL 4.1.

`-r, --replace`
: As opções `--replace` e `--ignore` controlam o tratamento de registros de entrada que duplicam registros existentes em valores de chaves únicas. Se você especificar `--replace`, novos registros substituirão registros que tiverem o mesmo valor na chave unica. Se você especificar `--ignore`, registros de entrada que duplicariam um registro existente em um valor de chave única são saltados. Se você não especificar nenhuma das duas opções, um erro ocorrerá quando um valor de chave duplicado for encontrado e o resto do arquivo texto será ignorado.

`-s, --silent`
: Modo silencioso. Gera saída somente quando ocorrer algum erro.

`-S /path/to/socket, --socket=/path/to/socket`
: O arquivo socket para usar ao conectar à `localhost` (máquina padrão).

`-u user_name, --user=user_name`
: O nome de usuário MySQL para usar ao conectar ao servidor. O valor padrão é seu nome de usuário atual no Unix.

`-v, --verbose`
: Modo verbose. Gera mais informações na saída.

`-V, --version`
: Exibe informação sobre a versão e sai.

Abaixo um exemblo da utilização de `mysqlimport`:

```
$ mysql --version
mysql  Ver 9.33 Distrib 3.22.25, for pc-linux-gnu (i686)
$ uname -a
Linux xxx.com 2.2.5-15 #1 Mon Apr 19 22:21:09 EDT 1999 i586 unknown
$ mysql -e 'CREATE TABLE imptest(id INT, n VARCHAR(30))' test
$ ed
a
100     Max Sydow
101     Count Dracula
.
w imptest.txt
32
q
$ od -c imptest.txt
0000000   1   0   0  \t   M   a   x       S   y   d   o   w  \n   1   0
0000020   1  \t   C   o   u   n   t       D   r   a   c   u   l   a  \n
0000040
$ mysqlimport --local test imptest.txt
test.imptest: Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
$ mysql -e 'SELECT * FROM imptest' test
+------+---------------+
| id   | n             |
+------+---------------+
|  100 | Max Sydow     |
|  101 | Count Dracula |
+------+---------------+
```

4.9.10 `mysqlshow`, Exibindo Bancos de Dados, Tabelas e Colunas

`mysqlshow` pode ser usado para exibir rapidamente quais bancos de dados existem, suas tabelas, e o nome das colunas da tabela.

Como o programa `mysql` você pode obter as mesmas informações com comandos `SHOW`. Veja Seção 4.6.8 [SHOW], Página 302.

`mysqlshow` é chamado assim:

```
shell> mysqlshow [OPÇÕES] [banco_dados [tabela [coluna]]]
```

- Se nenhum banco de dados é fornecido, todos os bancos de dados encontrados são exibidos.
- Se nenhuma tabela é fornecida, todas as tabelas encontradas no banco de dados são exibidas.
- Se nenhuma coluna for fornecida, todas colunas e tipos de colunas encontrados na tabela são exibidos.

Note que em versões mais novas do MySQL, você só visualiza as tabelas/bancos de dados/colunas para quais você tem algum privilégio.

Se o último argumento conter uma shell ou um meta-caracter do SQL, (`*`, `?`, `%` ou `_`) somente o que coincidir com o meta-caracter é exibido. Se um banco de dados conter underscore (`_`), eles devem ser precedidos por uma barra invertida (algumas shells de Unix irão exigir duas), para se obter tabelas/colunas apropriadamente. '*' são convertidos em metacaracteres '%' do SQL e '?' em metacaracteres '_' do SQL. Isto pode causar alguma confusão quando alguém tentar exibir as colunas para uma tabela com um `_`, neste caso o `mysqlshow` exibe somente os nomes de tabelas que casarem com o padrão. Isto é facilmente corrigido adicionando um `%` extra na linha de comando (como um argumento separador).

4.9.11 `mysql_config`, Opções para compilação do cliente MySQL

`mysql_config` lhe fornece informação útil sobre como compilar o seu cliente MySQL e conectá-lo ao MySQL.

`mysql_config` suporta as seguintes opções:

`--cflags` Parâmetros de compilação para encontrar arquivos incluídos e parâmetros e definições de compiladores criticos usados ao compilar a biblioteca `libmysqlclient`.

`--include`
Opções de compilador para encontrar arquivos de inclusão do MySQL. (Normalmente se usaria `--cflags` em vez disto)

`--libs` Bibliotecas e opções exigidas para ligar com a biblioteca cliente do MySQL.

`--libs_r` Bibliotecas e opções exigidas para ligar a biblioteca cliente do MySQL segura com thread.

`--socket` O nome socket padrão, definido ao configurar o MySQL.

`--port` O número da porta padrão, definida ao configurar o MySQL.

`--version`
Número da versão da distribuição MySQL.

`--libmysqld-libs ou --embedded`
Bibliotecas e opções exigidas para ligar com o servidor embutido MySQL.

Se você executar `mysql_config` sem nenhuma opção ele exibirá todas as opções suportadas mais os valores de todas elas:

```
shell> mysql_config
Usage: /usr/local/mysql/bin/mysql_config [OPTIONS]
Options:
        --cflags         [-I/usr/local/mysql/include/mysql -mcpu=pentiumpro]
--include        [-I/usr/local/mysql/include/mysql]
--libs           [-L/usr/local/mysql/lib/mysql -lmysqlclient -lz -lcrypt -lnsl -lm
--libs_r         [-L/usr/local/mysql/lib/mysql -lmysqlclient_r -lpthread -lz -lcryp
--socket         [/tmp/mysql.sock]
--port           [3306]
--version        [4.0.16]
--libmysqld-libs [-L/usr/local/mysql/lib/mysql -lmysqld -lpthread -lz -lcrypt -lnsl
```

Você pode usá-lo para compilar o cliente MySQL como a seguir:

```
CFG=/usr/local/mysql/bin/mysql_config
sh -c "gcc -o progname `$CFG --cflags` progname.c `$CFG --libs`"
```

4.9.12 `perror`, Explicando Códigos de Erros

Para a maioria dos erros de sistema o MySQL irá, em adição a uma mensagem de texto interna, imprimir também o código de erro do sistema em um dos seguintes estilos: `message ... (errno: #)` ou `message ... (Errcode: #)`.

Você pode descobrir o que o código de erro significa exeminando a documentação para o seu sistema ou usar o utilitário `perror`.

`perror` exibe a descrição para um código de erro do sistema, ou um código de erro do mecanismo de armazenamento MyISAM/ISAM (handler de tabela).

`perror` é utilizado assim:

```
shell> perror [OPÇÕES] [CÓDIGO_ERRO [CÓDIGO_ERRO...]]

Exemplo:

shell> perror 13 64
Error code  13:  Permission denied
Error code  64:  Machine is not on the network
```

Note que a mensagem de erro sã ona maioria dependente do sistema!

4.9.13 Como Executar Comandos SQL a Partir de um Arquivo Texto

O cliente `mysql` normalmente é usado de maneira interativa, desta forma:

```
shell> mysql banco_dados
```

Entretanto, também é possível colocar seus comandos SQL em um arquivo e dizer ao `mysql` para ler a entrada a partir deste arquivo. Para fazer isto, crie um arquivo texto '`arquivo_texto`' contendo os comandos que você deseja executar. Então execute o `mysql` como exibido abaixo:

```
shell> mysql banco_dados < arquivo_texto
```

Você também pode iniciar seu arquivo texto com uma instrução `USER nome_bd`. Neste caso, não é necessário especificar o nome do banco de dados na linha de comando:

```
shell> mysql < arquivo_texto
```

Se você já está executando o `mysql`, você pode executar um arquivo de script SQL usando o comando `source`:

```
mysql> source filename;
```

Para mais informações sobre o modo batch, Seção 3.5 [Modo batch], Página 193.

4.10 Os Arquivos de Log do MySQL

O MySQL tem vários arquivos de log diferentes que podem ajudá-lo a descobrir o que está acontecendo dentro do `mysqld`:

Log file	Description
O log de erros	Problemas encontrados iniciando, executando ou parando o `mysqld`.
O log isam	Documenta todas alterações a tabelas ISAM. Usado somente para depuração do código isam.
O log de consultas	Conexões estabelecidas e consultas executadas.
O log de atualizações	Desatulizado: Armazena todas as instruções que alteram dados.
O log binário	Armazena todas as instruções que alteram qualquer coisa. Usada também para replicação.
O log para consultas lentas	Armazena todas queries que levaram mais de `long_query_time` segundos para executar ou que não usaram índices.

Todos logs podem ser encontrados no diretório de dados do `mysqld`. Você pode forçar o `mysqld` a reabrir os arquivos de log (ou em alguns casos trocar para um novo log) executando `FLUSH LOGS`. Veja Seção 4.6.4 [FLUSH], Página 299.

4.10.1 O Log de Erros

A arquivo de log de erro contém informações indicando quando o `mysqld` foi iniciado e finalizado e também qualquer erro crítico encontrado na execução.

Se o `mysqld` finaliza inesperadamente e o `mysqld_safe` precisar reiniciar o `mysqld`, `mysqld_safe` gravará uma linha `restarted mysqld` neste arquivo. Este log também guarda um aviso se o `mysqld` notificar uma tabela que precisa ser automaticamente verificada ou reparada.

Em alguns sistemas operacionais, o log de erro irá conter registros de pilha de onde o `mysqld` finalizou. Isto pode ser usado para saber onde e como o `mysqld` morreu. Veja Seção E.1.4 [Utilizando registros de pilha], Página 1082.

A partir do MySQL 4.0.10 você pode especificar onde o `mysqld` armazena o arquivo de log de erro com a opção `--log-error[=filename]`. Se nenhum nome de arquivo for dado, o `mysqld` usará `mysql-data-dir/'maquina'.err` no Unix e '`\mysql\data\mysql.err`' no Windows.i Se você executar `flush logs` o arquivo antigo terá o prefixo `--old` e o `mysqld` criará um novo arquivo de log vazio.

Em versões mais antigas do MySQL o tratamento do log de erro era feito pelo `mysqld_safe` o qual redirecionava o arquivo de erro para `'maquina'.err`. Pode se alterar este nome de arquivo com a opção `--err-log=nome_arq`.

Se você não especificar `--log-error` ou se você utilizar a opção `--console`, o erro será escrito em stderr (o terminal).

No Windows a saída é sempre feita no arquivo `.err` se `--console` não for utilizado.

4.10.2 O Log de Consultas

Se você deseja saber o que acontece com `mysqld`, você deve iniciá-lo com a opção `--log[=arquivo]`. Isto irá documentar todas conexões e consultas no arquivo log (por padrão nomeado '`'nome_máquina'.log`'). Este log pode ser muito útil quando você suspeitar de um erro em um cliente e deseja saber exatamente o que o `mysqld` acha que o cliente enviou.

Older versions of the `mysql.server` script (from MySQL 3.23.4 to 3.23.8) pass `mysqld_safe` a `--log` option (enable general query log). If you need better performance when you start using MySQL in a production environment, you can remove the `--log` option from `mysql.server` or change it to `--log-bin`. Veja Seção 4.10.4 [Binary log], Página 374.

Versões mais antigas do script `mysql.server` (MySQL 3.23.4 a 3.23.8) passam ao `safe_mysql` uma opção `--log` (habilita a log de consulta geral). Se você precisar melhorar a performance quando iniciar o uso do MySQL em um ambiente de produção, pode remover a opção `--log` do `mysql.server` ou alterá-lo para `--log-bin`. Veja Seção 4.10.4 [Log binário], Página 374.

As entradas neste log são escritas quando o `mysqld` recebe as questões. Pode estar diferente da ordem em que as instruções são executadas. Isto está em contraste com o log de atualizações e o log binário nos quais as consultas são escritas depois de serem executadas, mas que quaisquer travas sejam liberadas.

4.10.3 O Log de Atualizações

NOTA: O log de atualizações está obsoleto e foi substituído pelo log binário. Veja Seção 4.10.4 [Log binário], Página 374. O log binário pode fazer qualquer coisa que poderia ser feito com o log de atualizações, e mais. **O log de atualização será removido no MySQL 5.0**

Quando iniciado com a opção `--log-update[=nome_arquivo]`, o `mysqld` grava um arquivo log contendo todos os comandos SQL que atualizam dados. Se nenhum arquivo for fornecido, o nome da máquina é usado. Se um nome de arquivo for fornecido, mas não possuir o caminho, o arquivo é gravado no diretório de dados. Se '`nome_arquivo`' não possuir uma extensão, o `mysqld` irá criar os arquivos com os nomes desta forma: '`nome_arquivo.###`', onde `###` é um número que é incrementado cada vez que `mysqladmin refresh` , `mysqladmin flush-logs` ou a instrução `FLUSH LOGS` forem executados ou o servidor for reiniciado.

NOTA: Para o esquema acima funcionar, você não pode criar seus próprios arquivos com o mesmo nome que os do log de atualização `+` algumas extensões que podem ser tratadas como números, no diretório usado pelo log de atualização!

Se forem utilizadas as opções `--log` ou `-l`, o `mysqld` escreve um log geral com o nome de arquivo '`nome_máquina.log`', e o reinicio e a recarga não geram um novo arquivo de log (embora ele seja fechado e reaberto). Neste caso você pode copiá-lo (no Unix) usando:

```
mv nome_máquina.log nome_máquina-antigo.log
mysqladmin flush-logs
cp nome_máquina-antigo.log para-diretório-backup
rm nome_máquina-antigo.log
```

O log de atualização é inteligente pois registra somente instruções que realmente alteram dados. Portanto, um `UPDATE` ou um `DELETE` com uma cláusula `WHERE` que não encontre nenhum registro não é escrito no log. Ele salta até instruções `UPDATE` que atribui a uma coluna o mesmo valor que ela possuia.

O registro da atualização é feito imediatamente após uma consulta estar completa mas antes que as bloqueios sejam liberados ou que algum commit seja feito. Isto garante que o log seja escrito na ordem de execução.

Se você desejar atualizar um banco de dados a partir de arquivos de logs de atualização, você pode fazer o seguinte (assumindo que seus logs de atualização estejam nomeados na forma '`nome_arquivo.###`'):

```
shell> ls -1 -t -r nome_arquivo.[0-9]* | xargs cat | mysql
```

`ls` é utilizado para obter todos os arquivos de log na ordem correta.

Isto pode ser útil se você tiver que recorrer a arquivos de backup depois de uma falha e desejar refazer as atualizações que ocorreram entre a hora do backup e a falha.

4.10.4 O Log Binário

O log binário deve substituiu o log de atualizações. O log de atualizações será removido do MySQL 5.0. O log binário contém toda informação que está disponível no log de atualizações em um formato mais eficiente e de maneira transacionalmente segura.

O log binário, como o antigo log de atualização, apenas registra instruções que realmente atualizam os dados. Assim um `UPDATE` ou um `DELETE` com um `WHERE` que não encontra nenhum registro não é gravado no log. Ele ignora mesmo instruções `UPDATE` que definam a uma coluna um valor que ela já tenha.

O propósito principal do log binário é poder atualizar o banco de dados durante uma operação de restauração de forma mais completa possível, já que o log binário conteria todas as atualizações feitas depois que um backup foi realizado.

O log binário é também usado para replicar um `mysqld` slave a partir de um master. Veja Seção 4.11 [Replicação], Página 378.

O log binário também contém informação sobre o tempo que cada consulta leva para atualizar o banco de dados. Ele não contém consultas que não modificam dados. Se você quiser registrar todas as consultas (por exemplo, para encontrar um consulta com problema) você deve usar o log geral de consultas. Veja Seção 4.10.2 [Query log], Página 373.

Quando iniciado com a opção `--log-bin[=nome_arquivo]`, o `mysqld` escreve um arquivo de log contendo todos comandos SQL que atualizam dados. Se nenhum arquivo for fornecido, ele aponta para o nome da máquina seguido de `-bin`. Se for fornecido o nome do arquivo, mas ele não tiver o caminho, o arquivo é escrito no diretório de dados.

Se você fornecer uma extensão à `--log-bin=nome_arquivo.extensão`, a extensão será removida sem aviso.

O `mysqld` irá acrescentar uma extensão ao nome de arquivo do log binário que é um número que é incrementado cada vez que `mysqladmin refresh`, `mysqladmin flush-logs`, a instrução `FLUSH LOGS` forem executados ou o servidor for reiniciado. Um novo log binário também será automaticamente criado quando o tamanho do log atual alcançar `max_binlog_size`. Nota se você estiver usando transações: uma transação é escrita em um bloco no arquivo de log binário, já que ele nunca é separado entre diversos logs binários. Desta forma, se você tiver grnades transações, você pode ter logs binários maiores que `max_binlog_size`.

Você pode deletar todos os arquivos de log binário com o comando `RESET MASTER` (veja Seção 4.6.5 [`RESET`], Página 301), ou apenas alguns deles com `PURGE MASTER LOGS` (veja Seção 4.11.7 [Replicação SQL], Página 400).

Você pode utilizar as seguintes opções ao `mysqld` para afetar o que é documentado pelo log binário (tenha certeza de ler as notas que seguem esta tabela):

Opção	**Descrição**
`binlog-do-db=nome_banco_dados`	Diz ao master que ele deve registrar atualizações no log binário se o banco de dado atual (ex.: aquele selecionado por `USE`) é 'nome_banco_dados'. Todos os outros bancos de dados que não forem explicitamente mencionados são ignorados. Note que se você utilizá-lo você deve se assegurar que você só faz atualizações no banco de dados atual. (Exemplo: `binlog-do-db=algum_bancodados`) Exemplo do que não funciona como você poderia esperar: se o servidor é iniciado com `binlog-do-db=sales`, e você fizer `USE prices; UPDATE sales.january SET amount=amount+1000;`, esta consulta não será gravada no log binário.
`binlog-ignore-db=nome_banco_dados`	Diz ao master que atualizações onde o banco de dados atual (ex.: aquele selecionado com `USE`) é 'nome_banco_dados' não deve ser gravado no log binário. Note que se você usar esta opção você deve ter certeza que você só faz atualizações no banco de dados atual. (Exemplo: `binlog-ignore-db=algum_banco_dados`) Exemplo do que não funciona como você poderia esperar: se o servidor é iniciado com `binlog-do-db=sales`, e você fizer `USE prices; UPDATE sales.january SET amount=amount+1000;`, esta consulta será gravada no log binário.

As regras estão avaliadas na seguinte ordem, para decidir se a consulta deve ser escrita no log binário ou não:

1. Existem as regras `binlog-do-db` ou `binlog-ignore-db`?
   - Não: grave a consulta no log binário e saia.

   - Sim: Vá para o passo abaixo.
2. Então existe algumas regras (`binlog-do-db` ou `binlog-ignore-db` ou ambos). Existe um banco de dados atual (algum banco de dados foi selecionado com `USE`?)?
   - Não: **NÃO** grave a consulta e saia.
   - Sim: vá para o passo abaixo.
3. Existe um banco de dados. Existe alguma regra `binlog-do-db`?
   - Sim: O banco de dados atual se encaixa em qualquer uma das regras `binlog-do-db`?
     - Sim: grave a consulta e saia.
     - Não: **NÃO** grave a consulta e saia.
   - Não: Vá para o passo abaixo.
4. Existem algumas regras `binlog-ignore-db`. O banco de dados atual se encaixa em qualquer uma das regras `binlog-ignore-db`?
   - Sim: não grave a consulta e saia.
   - Não: grave a consulta e saia.

Então, por exemplo, um slave em execução com apenas `binlog-do-db=sales` não gravará no log binário qualquer consulta em que o banco de dados atual é diferente de `sales` (em outras palavras, `binlog-do-db` pode, significar algumas vezes, "ignore outros bancos de dados").

Para saber quais arquivos binários foram usados, o `mysqld` irá criar também um arquivo de índice para o log binário que contém o nome de todos os arquivos de log binário usados. Por padrão este arquivo tem o mesmo nome que o arquivo de log binário, com a extensão `'.index'`. Você pode alterar o nome do arquivo de índice do log binário com a opção `--log-bin-index=[nome_arquivo]`. Você não deve eduitar este arquivo manualmente enquanto o `mysqld` estiver em execução; fazer isto confundiria o `mysqld`.

Se estiver sendo usado replicação, os arquivos de log binário antigos não devem ser apagados até ter certeza que nenhum slave irá mais precisar deles. Uma forma de fazer isto é o utilizar `mysqladmin flush-logs` uma vez por dia e então remover qualquer log com mais de 3 dias. Você pode removê-los manualmente, ou de preferência usando `PURGE MASTER LOGS` (veja Seção 4.11.7 [Replicação SQL], Página 400) o qual atualizará de forma segura o arquivo de índice do log binário para você (e que pode ter um argumento de data desde o MySQL 4.1)

Uma conexão com o privilégio `SUPER` pode desabilitar o registro no log binário de suas consultas usando `SET SQL_LOG_BIN=0`. Veja Seção 4.11.7 [Replicação SQL], Página 400.

Você pode examinar o arquivo de log binário com o utilitário `mysqlbinlog`. Por exemplo, você pode atualizar um servidor MySQL a partir de um log binário como mostrado a seguir:

```
mysqlbinlog arquivo-log | mysql -h nome_servidor
```

Veja Seção 4.9.5 [mysqlbinlog], Página 358 para mais informações sobre o utilitário `mysqlbinlog` e como utilizá-lo.

`mysqlbinlog --help` irá lhe fornecer mais informações de como usar este programa!

Se você estiver utilizando `BEGIN [WORK]` ou `SET AUTOCOMMIT=0`, você deve utilizar o log binário do MySQL para backups no lugar do antigo log de atualização.

O Log binário é feito imedatamente depois que uma consulta terminar mas antes que os bloqueios sejam liberados ou algum commit seja feito. Isto garante que o log seja feito na ordem de execução.

Atualizações em tabelas não transacionais são armazenadas o log binário imediatamentedepois da execução. Para tabelas tranascionais como `BDB` ou `InnoDB`, Todas atualizações (`UPDATE`, `DELETE` ou `INSERT`) que alteram uma tabela transacional são armazenadas no cache até um `COMMIT`. Quaisquer atualizações a uma tabela não transacional são armazenadas no log binário de uma vez. Todas as threads irão, no início, alocar um buffer de `binlog_cache_size` para registrar consultas. Se uma conaulta é maior que o registro, a thread irá criar um arquivo temporário para lidar com a mesma. O arquivo temporário será apagado quando a thread terminar.

O `max_binlog_cache_size` (padrão 4G) pode ser usado para restringir o tamanho total usado para armazenar uma consulta multi-transacional. Se uma transação é maior que isto ela falhará e fará um roll back.

Se você estiver utilizando o log de atualização ou o binário, inserções concorrentes não funcionarão juntas com `CREATE ... INSERT` e `INSERT ... SELECT`. Isto é para garantir que você possa recriar uma cópia exata de suas tabelas aplicando o log em um backup.

4.10.5 O Log para Consultas Lentas

Quando iniciado com a opção `--log-slow-queries[=file_name]` o `mysqld` escreve em um arquivo log contendo todos os comandos SQL que levam mais de `long_query_time` segundos para executar. O tempo para obter os bloqueios de tabelas iniciais não são contados como tempo de execução.

O log de consultas lentas é gerado depois que uma query é executada e depois de todas as bloqueios serem liberados. Ela pode estar em ordem diferente da que as instruções foram executadas.

Se nenhum nome de arquivo for fornecido, o padrão é o nome da máquina com o sufixo `-slow.log`. Se um nome de arquivo for especificado, mas não conter o caminho, o arquivo é gravado no diretório de dados.

O log para queries lentas pode ser usado para encontrar queries que levam muito tempo para executar e que devem ser candidatas a otimização. Com um log muito grande, isto pode ser uma tarefa difícil. Você pode utilizar o log de consultas lentas através do comando `mysqldumpslow` para obter um resumo das consultas que aparecem no log.

Se a opção `--log-long-format` estiver sendo usada, então as consultas que não estiverem utilizando índices serão escritas. Veja Seção 4.1.1 [Opções de linha de comando], Página 207.

4.10.6 Manutenção do Log de Arquivo

O MySQL tem vários arquivos de log que possibilitam ver o que está ocorrendo com mais facilidade. Veja Seção 4.10 [Arquivos de Log], Página 372. Porém de tempos em tempos deve ser feita uma limpeza nos arquivos de logs do `MySQL` para que eles não ocupem muito do espaço do disco.

Ao utilizar o MySQL com arquivos log, você necessitará de tempos em tempos remover antigos arquivos de log e dizer ao MySQL para logar com novos arquivos. Veja Seção 4.5.1 [Backup], Página 275.

Em uma instalação Linux `RedHat`), você pode usar o script `mysql-log-rotate` para isto. Se você instalou o MySQL de uma distribuição RPM, o script deve ter sido instalado automaticamente. Perceba que você deve ter cuidado com este script se você estiver utilizando o log binário para replicação!

Em outros sistemas você deve instalar um pequeno script que será executado pelo `cron` para lidar com os arquivos de log.

Você pode forçar o MySQL a iniciar utilizando novos arquivos de log usando `mysqladmin flush-logs` ou utlizando o comando SQL `FLUSH LOGS`. Se você usa o MySQL Versão 3.21 deve utilizar o comando `mysqladmin refresh`.

O comando acima faz o seguinte:

- Se o log padrão (`--log`) ou log de consultas lentas (`--log-slow-queries`) forem utilizados, fecha e reabre o arquivo de log. (‘`mysql.log`’ e ‘`‘hostname‘-slow.log`’ como padrão).
- Se o log de atualização (`--log-update`) é usado, fecha o log de atualização e abre um novo arquivo log com uma sequência numérica mais alta.

Se você só estiver utilizando o log de atualização, você tem apenas que atualizar os logs e então mover os arquivos de log antigos para um backup. Se você estiver utilizando o log normal, você pode fazer algo assim:

```
shell> cd diretório-dados-mysql
shell> mv mysql.log mysql.old
shell> mysqladmin flush-logs
```

e então fazer um backup e remover o ‘`mysql.old`’.

4.11 Replicação no MySQL

Capacidades de replicação permitidindo que os bancos de dados em um servidor MySQL seja duplicado em outro foram introduzidos no MySQL versão 3.23.15. Esta seção descreve os vários recursos da replicação no MySQL. Ele serve como uma referência para as opções disponíveis na replicação. Você será introduzido a replicação e aprenderá como implementá-la. Em direção ao final, existem algumas questões mais perguntadas (FAQ), descrições de problemas e como resolvê-los.

Sugeriemos que você visite nosso website em `http://www.mysql.com/` frequentemente e leia as atualizações desta seção. A replicação esta constantemente sendo melhorada e nós atualizamos o manual frequentemente com a informação mais atual.

4.11.1 Introdução

A partir da versão 3.23.15, o MySQL suporta replicação de uma via internamente. Um servidor atua como o master, enquando o outro atua como slave. O servidor master mantêm um log binário de atualizações (veja Seção 4.10.4 [Log binário], Página 374). É mantido também um arquivo de índices dos logs binários para manter os registro da rotatividade

dos logs. Cada slave, na conexão, informa ao master onde parou desde a última atualização propagada com sucesso, realiza a atualização e então para e espera o master informar sobre novas atualizações.

Um slave também pode ser um master se você condigurar uma cadeia de servidores em replicação.

Note que se você estiver usando replicação, todas atualizações nas tabelas replicadas devem ser realizadas no servidor master. Senão, você sempre deve ter cuidados para evitar conflitos entre as atualizações que os usuários enviam ao master e aquelas que os usuários enviam ao slave.

Replicação de uma via trazem benefícios de robustez, velocidade e administração do sistema:

- A robustez é aumentada com uma configuração master/slave. No evento de problemas com o master, você pode trocar para o slave como um backup.
- A velocidade extra é alcançada dividindo a carga das consultas dos clientes em processamento para entre os servidores master e slave, resultando em melhor tempo de resposta. Consultas `SELECT` podem ser enviadas para o slave para reduzir a carga do processamento das consultas do master. Consultas que modificam dados devem ainda ser enviados para o master e slave para não ficarem fora de sincronia. Esta estratégia de balancemento de carga é efetiva se consultas que não sejam de atualização dominarem, mas este é o caso normal.
- Outro benefício de utilizar replicação é que pode-se obter backups intantâneos do sistema fazendo backups no slave em vez de fazê-los no master. Veja Seção 4.5.1 [Backup], Página 275.

4.11.2 Visão Geral da Implementação da Replicação

A replicação no MySQL baseia-se no fato do servidor master manter o registro de todas as alterações de seus bancos de dados (atualizações, deleções, etc) no log binário. (veja Seção 4.10.4 [Log binário], Página 374). Cada servidor slave recebe do master consultas salvas no log binário, para que assim execute as mesmas consultas nos seus dados replicados.

É **muito importante** entender que o log binário é simplesmente um registro iniciando a partir de um ponto fixo no tempo (o momento que você habilitou o log binário). Quaisquer slaves que você configure necessitará de cópias do banco de dados do seu master como eles existiam no momento em que o log binário foi habilitado no master. Se você iniciar os slaves com dados diferentes daqueles do master **quando o log binário foi iniciado**, seus slaves falharão.

A seguinte tabela indica a compatibilidade de replicação master/slave entre diferentes versões do MySQL.

		Master 3.23.33 e posterior	**Master 4.0.0**	**Master 4.0.1**	**Master 4.0.3 e posterior**
Slave	**3.23.33 e posterior**	sim	não	não	não
Slave	**4.0.0**	não	sim	não	não
Slave	**4.0.1**	sim	não	sim	não
Slave	**4.0.3 e posterior**	sim	não	não	sim

Como regra geral, sempre é recomendado usar versões MySQL recentes, porque as capacidades de replicação estão sendo continuamente melhoradas. Com relação a versão 4.0, recomendamos usar a mesma versão para o master e o slave, com exceção de que o 4.0.2 não é recomandado para replicação.

Note qye quando você atualiza um mestre do MySQL 3.23 para o MySQL 4.0 (ou 4.1) você não deve reiniciar a replicação usando o log binário antigo da versão 3.23, porque isto infelizmente deixa o slave 4.0 confuso. A atualização pode seguramente feita deste modo, assumindo que você tenha uma mestre 3.23 para atualizar e você tenha slaves 4.0:

1. Bloqueie todas as atualizações no mestre (`FLUSH TABLES WITH READ LOCK`).
2. Espere até que todos os slaves tenham buscados todas as alterações pelo master (use `SHOW MASTER STATUS` no master, e `SELECT MASTER_POS_WAIT()` nos slaves). Então execute `STOP SLAVE` nos slaves.
3. Finalize o MySQL no master e atualize o master para o MySQL 4.0.
4. Reinicie o MySQL no master. Grave o nome '`<name>`' do log binário mais recentemente criado do master. Você pode obter o nome dos arquivos executando `SHOW MASTER STATUS` no master. Então envie estes comando em cada slave:

```
mysql> CHANGE MASTER TO MASTER_LOG_FILE='<name>', MASTER_LOG_POS=4;
mysql> START SLAVE;
```

Se você também deve atualizar seus slaves da versão 3.23 para 4.0, você deve primeiro atualizar seus slaves: Desligue cada um, atualize-os e os reinicie. Então atualize o master como descrito.

A partir da versão 4.0.0, pode se usar `LOAD DATA FROM MASTER` para configurar um escrao. Esteja certo que `LOAD DATA FROM MASTER` funciona atualmente apenas se todas as tabelas no master são do tipo `MyISAM`. Além disso, estas instrução irão adquirir lock global de leitura, assim nenhuma escrita será possível enquanto as tabelas estão sendo transferidas do master. Quando implementarmos hot backup de tabelas sem lock (no MySQL 5.0), este lock global de leitura não será mais necessário.

Devido a estas limitações, recomendamos que você só use `LOAD DATA FROM MASTER` se o conjunto de dados de master for relativamente pequeno, ou se um lock de leitura prolongado no master é aceitável. Enquanto a velocidade atual do `LOAD DATA FROM MASTER` pode variar de sistema para sistema, uma boa regra do dedão de quanto tempo será necessário é considerar 1 segundo por 1 MB do arquivo de dados. Você ficará próximo da estimativa se tanto o master quanto o slave forem equivalentes a um Pentium 700 Mhz e estiverem conectado a uma rede de 100 MBits/s. É claro, esta é apenas uma estimativa grosseira da ordem de magnitude.

Uma vez que o slave foi configurado corretamente e está em execução, ele simplesmente conectará ao master e esperará por atualizações nos processos. Se o master for desligado ou o slave perder conectividade com seu master, ele tentará conectar periodicamente até conseguir reconectar e constinuar as atualizações. O intervalo de tentativa é controlado pela opção `--master-connect-retry`. O padrão é 60 segundos.

Cada slave mantêm registro de onde parou. O servidor master não tem conhecimento de quandos slaves existem ou quais estão atualizados em um determinado momento.

4.11.3 Detalhes de Implementação da Replicação

Três threads estão envolvidas na replicação: uma no master e duas no slave. Quando `START SLAVE` é executado, a thread de E/S é criada no slave. Ela se conecta ao master e pede pelo envio de seus logs binários. Então uma thread (chamada `Binlog dump` no `SHOW PROCESSLIST` no master) é criada no master para enviar estes logs binários. A thread de E/S lê o que o `Binlog dump` envia e simplesmente a copia para algum arquivo local no diretorio de dados do slave chamado relay logs. A última thread, a thread de SQL, é criada no slave; ela lê o relay logs e executa as consultas contidas nele.

Note que o master tem uma thread para cada servidor slave atualmente conectado.

Com `SHOW PROCESSLIST` você pode saber o que está acontecendo no master e no slave em relação a replicação.

O exemplo seguinte ilustra como as três threads aparecem em `SHOW PROCESSLIST`. O formato da saída é aquele usado por `SHOW PROCESSLIST` a partir do MySQL versão 4.0.15, quando o conteúdo da coluna `State` foi alterado para ser mais significativo comparado com versões alterações.

No servidor master a saída se parece com isto:

```
mysql> SHOW PROCESSLIST\G
*************************** 1. row ***************************
     Id: 2
   User: root
   Host: localhost:32931
     db: NULL
Command: Binlog Dump
   Time: 94
  State: Has sent all binlog to slave; waiting for binlog to be updated
   Info: NULL
```

No servidor slave, a saída se parece com isto:

```
mysql> SHOW PROCESSLIST\G
*************************** 1. row ***************************
     Id: 10
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 11
  State: Waiting for master to send event
   Info: NULL
*************************** 2. row ***************************
     Id: 11
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 11
```

```
      State: Has read all relay log; waiting for the slave I/O thread to update it
       Info: NULL
```

Aqui a thread 2 está no master. A thread 10 é a thread de E/S no slave. A thread 11 é a thread de SQL no slave; note que o valor na coluna `Time` pode dizer quando o slave é comparado com o master (veja Seção 4.11.9 [FAQ da Replicação], Página 410).

A lista a seguir mostra os estados mais comuns que você verá na coluna `State` para a thread `Binlog Dump` do master. Se você não ver estas threads em um servidor master, a replicação não está sendo executada.

`Sending binlog event to slave`
: Logs binários consistem de eventos, onde um evento é normamente uma consulta mais alguma informação. A thread lê um evento do log binário e ele é enviado para o slave.

`Finished reading one binlog; switching to next binlog`
: A thread finalizou a leitura de um log binário e está abrindo o seguinte a ser enviado para o slave.

`Has sent all binlog to slave; waiting for binlog to be updated`
: A thread leu todos os log binários e está inativa. Ela está esperando por conexões no master para gravar mais dados no log binário, se ele quiser.

`Waiting to finalize termination`
: Estado muito breve que ocorre quando a thread para.

Aqui estão os estados mais comuns que você verá na coluna `State` para a thread de E/S de um servidor slave. A partir do MySQL 4.1.1, este estado também aparece na coluna `Slave_IO_State` da saída de `SHOW SLAVE STATUS`. Isso significa que você pode ter uma boa visão do que está acontecendo apenas com `SHOW STATUS SLAVE`.

`Connecting to master.`
: Conectando ao master.

`Checking master version.`
: Estado muito breve que ocorre um pouco depois da conexão ser estabelecida.

`Registering slave on master.`
: Estado muito breve que ocorre um pouco depois da conexão ser estabelecida.

`Requesting binlog dump.`
: Estado muito breve que ocorre um pouco depois da conexão com o master ser estabelecida. A thread envia ao master um pedido para envio do conteúdo de seu log binário, iniciando a partir do log binário requisitado e sua posição.

`Waiting to reconnect after a failed binlog dump request.`
: Se o pedido de dump do log binário falhar (devido a desconexão), a thread fica neste estado enquanto está inativa. A thread fica inativa por `master-connect-retry` segundos antes de uma nova tentativa.

`Reconnecting after a failed binlog dump request.`
: Então a thread tenta se conectar com o master.

`Waiting for master to send event.`
: A thread conectou e está esperando que os eventos do log binário cheguem. Isto pode demorar se o master estiver inativo. Se a espera for maior que `slave_read_timeout` segundos, o tempo se esgotará. Neste ponto, a thread irá considerar a conexão quebrada e fará uma nova tentativa de conexão.

`Queueing master event to the relay log.`
: A thread leu o evento e o está copiando para o ser relay log para que a thread SQL possa processá-lo

`Waiting to reconnect after a failed master event read.`
: Um erro ocorreu durante a leitura (devido a desconexão); inativo por `master-connect-retry` segundos antes de tentar se reconectar.

`Reconnecting after a failed master event read.`
: Então a thread tenta se reconectar. Quando a conexão é estabelecida novamente, o estado se tornará `Waiting for master to send event`.

`Waiting for the slave SQL thread to free enough relay log space`
: Você está usando um valor `relay_log_space_limit` diferente de zero e os relay logs tem crescido tanto que o seu tamanho combinado excedem este valor. A thread E/S então espera até que a thread SQL libere espaço suficiente deletando o conteúdo dos relay logs e assim poder deletar alguns arquivos de relay logs.

`Waiting for slave mutex on exit.`
: Estado muito breve que ocorre quando a thread esta parando.

Aqui estão os estado mais comuns que você verá na coluna `State` para a thread de SQL de um servidor slave:

`Reading event from the relay log`
: A thread leu um evento do relay log para poder processá-lo.

`Has read all relay log; waiting for the slave I/O thread to update it`
: A thread processou todos os eventos nos arquivos de relay logs e está esperando a thread de E/S gravar novos eventos no relay log.

`Waiting for slave mutex on exit.`
: Estado muito breve que ocorre quando a thread é parada.

A coluna `State` para a thread de E/S também podem mostrar um string de consulta. Isto indica que a thread leu um evento do relay log, extraiu a conulta dele e está a está executando.

Antes do MySQL 4.0.2, as threads de E/S e SQL eram combinadas em uma só e nenhum relay log era usado. A vantagem do uso de duas threads é que elas separam a leitura e a execução da consulta em duas tarefas independentes, e assim o trabalho de leitura da consulta não se torna lento se a execução da consulta for lento. Por exemplo, se o servidor slave não estiver em execução por um instante, a sua thread de E/S pode rapidamente buscar todos o conteúdo dos logs binários do master quando o slave iniciar, mesmo se a thread de SQL demorar e levar horas para pegar os logs. Se o slave parar antes da thread SQL executar todas as consultas buscadas, a thread de E/S terá finalmente buscado tudo e assim um cópia segura das consultas estará armazenada localmente nos relay logs do slave para

execução na próxima execução do slave. Isto permite que os log binários sejam apagados no master, já que não há mais necessidade de esperar que o slave busque o conteúdo deles.

Por padrão, relay logs são nomeados usando nome de arquivos da forma '`host_name-relay-bin.nnn`', onde `host_name` é o nome da máquina servidora slave e `nnn` é uma sequência numérica. Arquivos de relay logs sucvessivos são criados usando uma sequência de números sucessiva, começando com `001`. O slave mantém registro dos relay logs em uso atualmente em um arquivo de índice. O nome de arquivo padrão dos relay logs é '`host_name-relay-bin.index`'. Por padrão estes arquivos são criados no diretório de dados do slave. O nome de arquivo padrão pode ser sobrescrito com as opções `--relay-log` e `--relay-log-index` do servidor.

Relay logs têm o mesmo formato dos logs binários, assim ele podem ser lidos com `mysqlbinlog`. Um relay log é automaticamente deletado pela thread de SQL tão logo não seja mais necessária (ex.: assim que tiver sido executado todos os seus eventos). Não existem comandos para deletar relay logs já que a thread SQL cuida de fazê-lo. No entanto, a partir do MySQL 4.0.14, `FLUSH LOGS` rotaciona os relay logs), o que irá influenciar quando a thread de SQL deletá-los.

Um novo relay log é criado sob as seguintes condições:

- A primeira vez que a thread de E/S inicia depois que o servidor slave inicia (No MySQL 5.0, um novo relay log será criado a cada vez que a thread de E/S inicia, não apenas pela primeira vez.)
- Uma instrução `FLUSH LOGS` é executada (a partir da versão 4.0.14).
- O tamanho do relay log atual se torna muito grande. O significado de "muito grande" é determinado da seguinte forma:
  - `max_relay_log_size`, se `max_relay_log_size > 0`
  - `max_binlog_size`, se `max_relay_log_size = 0` ou o MySQL é mais velho que 4.0.14

Um servidor de replicação slave cria dois arquivos pequenos no diretório de dados. Estes arquivos são chamados '`master.info`' e '`relay-log.info`' por padrão. Eles possuem informação como aquela mostrada na saída da instrução `SHOW SLAVE STATUS` (veja Seção 4.11.8 [Replicação], Página 402 para uma descrição deste comando). Como imagem de discos, eles sobrevivem ao desligamento do slave. A próxima vez que o slave é reiniciado, ele pode ler estes arquivos para saber o quanto ele processou do log binário do master e do seus próprios relay logs.

O arquivo '`master.info`' é atualizado pela thread de E/S.

A correspondência entre as linhas do arquivo e as colunas mostradas por `SHOW SLAVE STATUS` aparece a seguir:

Linha	**Descrição**
1	`Master_Log_File`
2	`Read_Master_Log_Pos`
3	`Master_Host`
4	`Master_User`
5	Senha (não mostrado por `SHOW SLAVE STATUS`)
6	`Master_Port`
7	`Connect_Retry`

O arquivo '`relay-log.info`' é atualizada pela thread de SQL. A correspondência entre as linhas do arquivo e as colunas mostradas por `SHOW SLAVE STATUS` apaerece a seguir:

Linha	Descrição
1	`Relay_Log_File`
2	`Relay_Log_Pos`
3	`Relay_Master_Log_File`
4	`Exec_Master_Log_Pos`

Quando você faz backup dos dados de seu slave, você deve fazer backup destes 2 pequenos arquivos, junto com seus relay logs pois eles são necessários para continuar a replicação depois que você restaurar os dados do slave. Se você perder os seus relay logs mas ainda tiver o arquivo '`relay-log.info`', você pode verifclos para determinar por quanto tempo a thread de SQL executou no log binário do master. Então você pode usar `CHANGE MASTER TO` com as opções `MASTER_RELAY_LOG` e `MASTER_RELAY_POS` para dizer ao slave para reler os log binários a partir deste ponto. Isto exige que o log binário ainda exista no servidor master. é claro.

Se seu slave está sujeito a replicação de instruções `LOAD DATA INFILE`, você também deve fazer backup dos arquivos '`SQL_LOAD-*`' que podem existir no diretório que o slave utiliza para este propósito. O slave precisará destes arquivos para continuar a replicação de qualquer instrução `LOAD DATA INFILE` interrompido.

A localização do diretório é especificada usando a opção '`--slave-load-tmpdir`'. Seu valor padrão, se não especificado, é o valor da variável `tmpdir`.

4.11.4 Como Configurar a Replicação

Aqui está uma descrição rápida de como configurar uma replicação completa em seu servidor MySQL atual. Ele assume que você deseja replicar todos os bancos de dados e nunca configurou uma replicação anteriormente. Você precisará desligar seu servidor master rapidamente para completar os passos delineados abaixo.

O procedimento é gravado para a configuração de um único slave, mas você pode usá-lo para configurar vários slaves.

Este método é o modo mais direto de se configurar um slave, mas ele não é o único. Por exemplo, se você já tem uma cópia instantânea dos dados do master, e o master já tem o seu ID do servidor definido e o log binário habilitado, você pode configurar um slaver sem desligar o master ou mesmo bloquear suas atualizações. Para maiores detalhes, veja Seção 4.11.9 [Replication FAQ], Página 410.

Se você deseja administrar uma configuração de replicação MySQL, sugerimos que leia todo este capítulo e experimente todos os comandos mencionados em Seção 4.11.7 [Replication Master SQL], Página 400 e Seção 4.11.8 [Replication Slave SQL], Página 402. Você também deve se familiarizar com as opções de inicialização da replicação em '`my.cnf`' na Seção 4.11.6 [Replication Options], Página 392.

Note que este procedimento e algumas das instruções SQL da replicação em seções posteriores se referrem ao privilégio `SUPER`. Antes do MySQL 4.0.2, use o privilégio `PROCESS`.

1. Certifique-se que você possui uma versão recente do MySQL instalado no servidor master e no(s) slave(s), e que estas versões sào compatíveis de acordo com a tabela mostrada em Seção 4.11.2 [Replication Implementation], Página 379.

Por favor não relate os erros até que você tenha verificado que o problema está presente na última distribuição.

2. Configure uma conta no servidor master com o com a qual o slave possa se conectar. Deve ser dada a esta conta o privilégio `REPLICATION SLAVE`. (Se a versão do MySQL for anterior a 4.0.2, de à conta o privilégio `FILE`.) Se a conta é somente para a replicação (o que é recomendável), então você não precisará fornecer nenhum privilégio adicional para ele.

   O nome de máquina no nome da conta deve ser aquele usado por cada um dos servidores slaves para conectar ao master. Por exemplo, para criar um usuário chamado `repl` que pode acessar seu master de qualquer máquina, você deve utilizar este comando:

   ```
   mysql> GRANT REPLICATION SLAVE ON *.* TO repl@'%' IDENTIFIED BY '<password>';
   ```

   Para versões do MySQL anteriores a 4.0.2, use este comando:

   ```
   mysql> GRANT FILE ON *.* TO repl@'%' IDENTIFIED BY '<password>';
   ```

   Se você planeja usar as instruções `LOAD TABLE FROM MASTER` ou `LOAD DATA FROM MASTER` a partir da máquina slave, você precisará de permissão para esta conta adicional.

   - Conceda a conta os privilégios globais `SUPER` e `RELOAD`.
   - Conceda o privilégio `SELECT` em todas as tabelas que você deseja carregar. Qualquer das tabelas master nas quais a conta não possa fazer um `SELECT` serão ignoradas por `LOAD DATA FROM MASTER`.

3. Se você estiver usando tabelas MyISAM, descarregue todas as tabelas e bloqueie as consultas de escrita executando o comando `FLUSH TABLES WITH READ LOCK`

   ```
   mysql> FLUSH TABLES WITH READ LOCK;
   ```

   e faça uma cópia de todos os dados existentes em seu servidor master.

   A maneira mais fácil de fazer isto é simplesmente usar um programa (`tar` no Unix, `PowerArchiver`, `WinRAR`, `WinZip` ou qualquer outro software similar no Windows) para produzir um arquivo de banco de dados no diretório de dados do seu master. Por exemplo, para usar `tar` que cria um arquivo que inclui todos os bancos de dados, altere a localização no diretório de dados do servidor master, e então execute este comando:

   ```
   shell> tar -cvf /tmp/mysql-snapshot.tar .
   ```

   Se você quiser que o arquivo inclua apenas um banco de dados chamado `estebd`, utilize este comando:

   ```
   shell> tar -cvf /tmp/mysql-snapshot.tar ./this_db
   ```

   Então copie o arquivo para o diretório '`/tmp`' na máquina servidora slave. Naquela máquina, altere a localização em um diretório de dados do slave e desempacote o arquivo usando este comando:

   ```
   shell> tar -xvf /tmp/mysql-snapshot.tar
   ```

   Você pode não desejar replicar o banco de dados `mysql`. Se não, você pode excluí-lo do arquivo. Você também não precisa incluir qualqer arquivo de log nos arquivos '`master.info`' ou '`relay-log.info`'.

   Enquanto o lock de leitura colocado por `FLUSH TABLES WITH READ LOCK` estiver em funcionando, leia o valor atual do nome do log binário e offset no master:

   ```
   mysql > SHOW MASTER STATUS;
   ```

```
+---------------+----------+--------------+------------------+
| File          | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+---------------+----------+--------------+------------------+
| mysql-bin.003 | 73       | test,bar     | foo,manual,mysql |
+---------------+----------+--------------+------------------+
1 row in set (0.06 sec)
```

A coluna `File` exibe o nome do log, enquanto `Position` exibe o offset. No exemplo acima, o valor do log binário é `mysql-bin.003` e o offset é 73. Grave os valores. Você precisará usá-los mais tarde quando estiver configurando o slave.

Uma vez realizada a cópia e gravado o nome do log e offset, você pode reabilitar a atividade de escrita no master:

```
mysql> UNLOCK TABLES;
```

Se você estiver usando tabelas InnoDB, você deve usar a ferramente InnoDB Hot Backup que está disponível para aqueles que compraram as licenças comerciais do MySQL, suporte ou a própria ferramenta de backup. Ele faz uma cópia consistente sem fazer nenhum lock no servidor master, e grava o nome do log e o offset correspondente em um snapshot para ser usado postriormente no slave. Mais informações sobre esta ferramenta esta disponível em `http://www.innodb.com/order.php`.

Sem a ferramenta Hot Backup, o modo mais rápido para tirar uma cópia das tabelas InnoDB é desligar o servidor master e copiar os arquivos e logs de dados do InnoDB e os arquivos de definição de tabela (`.frm`). Para gravar o nome e offset do arquivo de log atual você deve fazer o seguinte antes de desligar o servidor:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SHOW MASTER STATUS;
```

E então grave o nome e offset do log da saída de `SHOW MASTER STATUS` como mostrado anteriormente. Uma vez gravado o nome e o offset do log, desligue o servidor sem destravar as tabelas para se certificar que ele finalizará com a cópia correspondente ao arquivo de log e offset:

```
shell> mysqladmin -uroot shutdown
```

Uma alternativa para tabelas MyISAM e InnoDB é fazer um dump SQL do master em vez de uma cópia binária como acima; para isso você pode usar `mysqldump --master-data` em seu master e mais tarde executar o dump SQL em seu slave. No entanto, isto é mais lento que fazer a cópia binária.

Se o master foi executado anteriormente sem o `--log-bin` habilitado, os valores do nome do log e da posição mostrados por `SHOW MASTER STATUS` ou `mysqldump` estarão vazios. Neste caso, grave a string vazia (") para o nome do log e 4 para o offset.

4. Assegure-se que a seção `[mysqld]` do arquivo '`my.cnf`' no master inclui a opção `log-bin`. Esta seção também deve conter a opção `server-id=unique number`, onde `master_id` deve ser um valor inteiro entre 1 e 2^32 - 1. Por exemplo:

```
[mysqld]
log-bin
server-id=1
```

Se estas opções não estão presentes, adicione-as e reinicie o servidor.

5. Pare o servidor que será usado como slave e adicione o seguinte ao arquivo '`my.cnf`':

```
[mysqld]
server-id=slave_id
```

O valor `slave_id`, como o valor `master_id`, deve ser um valor inteiro de 1 to 2^32 - 1. Adicionalmente, é muito importante que o ID do slave seja diferente do ID do master. Por exemplo:

```
[mysqld]
server-id=2
```

Se você estiver usando vários servidores, cada um deve ter um valor `server-id` que seja diferente daquele do master e de cada um dos slaves. Pense nos valores de `server-id` como algo similar ao endereço IP: Estes IDs identificam de forma única cada instância de servidor na comunidade dos parceiros de replicação.

Se você não quiser especificar um `server-id`, ele será configurado com 1 se você não tiver definido `master-host`, senão ele será definido com 2. Note que no caso de omissão do `server-id`, um master irá recusar conexões de todos os slaves e um slave irá recusar se conectar a um master. Assim, omitir `server-id` só é bom para backups com um log binário.

6. Se você fizer um backup biário dos dados do servidor master, copie-o para o diretório de dados do servidor slave antes de iniciá-lo. Certifique-se que os privilégios nos arquivos e diretórios estão corretos. O usuário com o qual o MySQL executa precisa estar apto a lê-los e alterá-los, assim como no master.

   Se você fizer um backup usando `mysqldump`, inicie o slave primeiro (veja o próximo passo).

7. Inicie o servidor slave. Se ele tiver sido replicado previamente, inicie o servidor slave com a opção `--skip-slave-start`. Você também pode querer iniciar o servidor slave com a opção `--log-warnings`. Deste modo você irá obter mais mensagens sobre problemas (por exemplo, problemas de rede, ou conexão).

8. Se você fez um backup dos dados do servidor master usando `mysqldump`, carregue o arquivo de dump no servidor slave:

   ```
   shell> mysql -u root -p < dump_file.sql
   ```

9. Execute os seguintes comandos no slave, substutitua os valores dentro de `<>` com o os valores atuais relevantes ao ser sistema:

   ```
   mysql> CHANGE MASTER TO
       ->     MASTER_HOST='<master host name>',
       ->     MASTER_USER='<replication user name>',
       ->     MASTER_PASSWORD='<replication password>',
       ->     MASTER_LOG_FILE='<recorded log file name>',
       ->     MASTER_LOG_POS=<recorded log offset>;
   ```

   A tabela a seguir lista o tamanho máximo da string para as variáveis:

`MASTER_HOST`	60
`MASTER_USER`	16
`MASTER_PASSWORD`	32
`MASTER_LOG_FILE`	255

10. Inicie a thread slave:

    ```
    mysql> START SLAVE;
    ```

Depois de realizado este procedimento, o slave deve se conectar ao master e pegar todas as atualizações que ocorreram desde que o backup foi restaurado.

Se você esqueceu de configurar o `server-id` no master, os slaves não poderão se conectar a eles:

Se você esqueceu de configurar o `server-id` no slave, você irá obter o seguinte erro no arquivo de log:

```
Warning: one should set server_id to a non-0 value if master_host is set.
The server will not act as a slave.
```

Você também encontrará mensagens de erro no log de erro do slave se ele não puder replicar por qualquer motivo.

Uma vez que um slave está replicando, você encontrará um arquivo chamado `master.info` e um chamado `relay-log.info` no diretório de dados. Estes dois arquivos são usados pelo slave para manter o registro de quanto foi processado do log binário do master. **Não** remova ou edite o arquivo, a menos que você realmente saiba o que está fazendo e entenda as implicações. Mesmo neste caso, é mais aconselhável usar o comando `CHANGE MASTER TO`.

NOTA: o conteúdo de '`master.info`' sobrepõe algumas opções especificadas na lina de comando ou no '`my.cnf`' veja Seção 4.11.6 [Opções de Replicação], Página 392 para mais detalhes.

Agora que você tem uma cópia instantânea, você pode usá-la para configurar outros slaves. Para isso siga a porção referente ao slave descrita acima. Você não precisa ter outra cópia do master.

4.11.5 Recursos de Replicação e Problemas Conhecidos

Abaixo uma explicação do que é e o que não é suportado:

- A Replicação será feita corretamente com valores `AUTO_INCREMENT`, `LAST_INSERT_ID` e `TIMESTAMP`.
- As funções `USER()` e `LOAD_FILE()` são replicadas sem alterações e não funcionarão de forma confiável no slave. Isto também é verdade para `CONNECTION_ID()` em versões de servidor slaves mais antigas que 4.1.1. A **nova** função `PASSWORD()` no MySQL 4.1, é bem replicada desde os masters 4.1.1; o seu slave deve ser 4.1.0 ou acima para replicá-la. Se você tem slaves mais antigos e precisa replicar `PASSWORD()` do seu master 4.1.x, você deve iniciar o master com a opção `--old-password`.
- As variávies `SQL_MODE`, `UNIQUE_CHECKS`, `SQL_SELECT_LIMIT`, `SQL_AUTO_IS_NULL` e `TABLE_TYPE` não são replicados ainda. `FOREIGN_KEY_CHECKS` é replicado desde a versão 4.0.14.
- Você deve uilizar o mesmo conjunto de caracteres (`--default-character-set`) no master e slave. Senão, você pode conseguir erros de chaves duplicadas no slave, pois uma chave que é considrada como única no conjunto de caracteres no master pode não ser único no conjunto de caracteres do slave.
- Se você estiver usando tabelas transacionais no master e não transacionais (para as mesmas tabelas) no slave, você terá prblemas se o slave for parado no meio de um bloco `BEGIN/COMMIT`, já que o slave irá, mais tarde, iniciar a partir do início do bloco `BEGIN`. Este assunto está em nosso TODO e será corrigido em um futuro próximo.

- Consultas de atualização que usam variáveis de usuários são mal replicadas nas versões 3.23 e 4.0. Isto é corrigido no MySQL 4.1. Note que nomes de variáveis de usuários são caso insensitivo a partir da versão 5.0, assim você deve levar isto em conta quando configurar uma replicação entre um servidor com versão 5.0 e outro com uma versão anterior.
- O slave pode se conectar ao master usando SSL, se o master e o slave forem ambos 4.1.1 ou mais novos.
- Embora nunca tenhamos tido casos de ocorrênciar reais, é teoricamente possível de que o dado no master e no slave podem estar diferentes se uma consulta é projetada de modo que a modificação do dado seja não determinística, p.ex. deixar a vontade do otimizados de consultas (o que geralmente não é uma boa prática, mesmo fora da replicação!). Para uma explicação detalhada Seção 1.8.6.2 [Open bugs], Página 54.
- Antes do MySQL 4.1.1, os comandos `FLUSH`, `ANALYZE`, `OPTIMIZE` e `REPAIR` não são armazenados no log binário e por isto não são replicados para o slave. Isto normalmente não é um problema já que estes comandos não alteram nada. Isto significa, no entanto, que se você atualizar a tabela de privilégio do MySQL diretamente sem usar a instrução `GRANT` e replicar o banco de dados de privilégios `mysql`, você deve fazer um `FLUSH PRIVILEGES` em seu slave para que os novos privilégios tenham efeito. Também, se você utilizar `FLUSH TABLES` ao renomear uma tabela `MyISAM` envolvida em uma tabela `MERGE`, você terá uqe executar `FLUSH TABLES` manualmente no servidor. Desde o MySQL 4.1.1, estes comandos são escritos no log binário (exceto `FLUSH LOGS`, `FLUSH MASTER`, `FLUSH SLAVE`, `FLUSH TABLES WITH READ LOCK`) a menos que você especifique `NO_WRITE_TO_BINLOG` (ou seu alias `LOCAL`). Para um exemplo da sintaxe Seção 4.6.4 [FLUSH], Página 299.
- O MySQL suporta somente um master e vários slaves. Posteriormente adicionaremos um algorítimo de votação para trocar automaticamente o master se alguma coisa estiver errada com o master atual. Iremos também introduzir processos agentes para ajudar a fazer o balanceamento de carga enviando consultas `SELECT` para diferentes slaves.
- Tabelas temporárias são replicadas, exceto no caso em que você desliga o servidor slave (e não apenas a thread slave), e você tem alguns tabelas temporárias replicadas e são usadas em instruções UPDATES que ainda não foram executadas no slave. (Se você desligar o slave, as tabelas temporárias necessárias por estas atualizações não estarão mais disponíveis quando o slave iniciar novamente.) Para evitar este problema, não desligue o servidor enquanto ele tiver tabelas temporárias abertas. Em vez disto, use este procedimento:
  1. Envie uma instrução `STOP SLAVE`.
  2. Use `SHOW STATUS` para verificar o valor da variável `Slave_open_temp_tables`.
  3. Se o valor é 0, envie um comando `mysqladmin shutdown` para desligar o slave.
  4. Se o valor é diferente de 0, reinicie as threads slaves com `START SLAVE`.
  5. Repita o procedimento anterior para ver se você terá melhor sorte na próxima vez.

  Planejamoc corrigir este problema em um futuro próximo.
- É seguro conectar servidores em um relacionamento master/slave circular com `log-slave-updates` habilitado. Note, entretanto, que várias consultas não irão funcionar corretamente neste tipo de configuração a menos que o código do cliente seja escrito para

tomar cuidado dos potenciais problemas que podem ocorrer em diferentes sequências em servidores diferentes.

Isto significa que você pode fazer uma configuração parecida com o seguinte:

```
A -> B -> C -> A
```

As IDs do servidor são codificadas nos eventos do log binário. A saberá quando o evento que ele lê é foi originalmente criado por A, assim A não o executará não haverá loop infinito. Mas esta configuração circular só funcionará se você realizar atualizações não conflitantes entre as tabelas. Em outras palavras, se você insere dados em A e C, você nunca deve inserir um registro em A que pode ter uma chave confiltante com um registro em C. Você também não deve atualizar os mesmos registros em dois servidores se a ordem que a atualização é aplicada importa.

- Se houver um erro em uma consulta no slave, a thread slave irá terminar e uma mensagem irá aparecer no log de erro do slave. Você deve então conectar a um slave manualmente, corrigir a causa do erro (por exemplo, tabela não existente), e então executar o comando sql `SLAVE START`.
- Se a conexão para o master for perdida, o slave irá tentar se reconectar imediatamente. Se ele falhar, o slave irá tenatr a cada `master-connect-retry` segundos (padrão 60). Por causa disto, é seguro desligar o master, e então reiniciá-lo depois de um tempo. O slave também está apto para lidar com interrupções de rede. No entanto o slave notificará a a perda da rede apenas após não ter recebido dados do master por `slave_net_timeout` segundos. Assim se sua perda for pequena, você pode querer diminuir `slave_net_timeout`. Veja Seção 4.6.8.4 [`SHOW VARIABLES`], Página 309.
- Desligar o slave (corretamente) também é seguro, pois mantém sinais de onde parou. Desligamentos incorretos podem produzir problemas, especialmente se o cache de disco não foi sincronizado antes do sistema morrer. Seu sistema de tolerância a falhas será melhorado se você possuir um bom No-Break ou UPS.
- Devido a natureza não trabnsacional das tabelas MyISAM, é possível ter uma consulta que atulizará apenas parcialmente uma taela e retornará um código de erro. Isto pode acontecer, por exemplo, em uma inserção multi-registro que tem uma violação da restrição da chave o use uma consulta de atualização é finalizada após atualizar alguns dos registros. Se isto acontecer no master, a thread slave sairá e irá esperar o DBA decidir o que fazer com isto a menos que seja autenticado e a execução da consulta resulte no mesmo código de erro. Se este comportamento da validação do código de erro não for desejável, algum (ou todos) os erros podem ser ignorados com a opção `--slave-skip-errors`. Ela está disponível a partir da versão 3.23.47.
- Se você atualiza tabelas transacionais a partir de tabelas não-transacioanis dentro de um segmento `BEGIN/COMMIT`, a atualização no log binário pode estar fora de sincronia se algumas threads alterarem a tabela não transacional antes do commit da transação. Isto é porque a transação é escrita no log binário apenas quando é feito o commit.
- Antes da versão 4.0.15, qualquer atualização de uma tabela não transacional é gravada no log binário imeditamente quando a atualização é feita enquanto atualizações transacionais são gravadas no `COMMIT` ou não gravadas se você utilizar um `ROLLBACK`. Você deve levar isto em conta quando atualizar tabelas transacionais e não transacionais na mesma transação e você estiver usando o log binário para backup ou replicação. Na versão 4.0.15 nós alteramos o comportamento do registro de transações que misturam

atualizações de tabelas transacionais e não transacionais, que soluciona o problema (ordem das consultas boas no log binário, e todas as consultas necessárias são gravadas no log binário mesmo no caso de um `ROLLBACK`). O problema que permanece é quando uma segunda conexão atualiza uma tabela não transacional enquanto a primeira transação da conexão ainda não está finalizada (ordenação errada ainda pode ocorrer, porque a atualização da segunda conexão será gravada imediatamente depois de ela ter sido feita).

A seguinte tabela lista problemas na versão 3.23 que estão corrigidas na versão 4.0:

- `LOAD DATA INFILE` é tratado apropriadamente desde que o arquivo ainda esteja no servidor master no momento da propagação da atualização.
- `LOAD LOCAL DATA INFILE` será ignorado.
- Na versão 3.23 `RAND()` em atualizações não é replicado apropriadamente. Use `RAND(alguma_expr_nao_rand)` se você estiver replicando atualizções com `RAND()`. Você pode, por exemplo, utilizar `UNIX_TIMESTAMP()` para o argumento de `RAND()`. Isto é corrigido na versão 4.0.

4.11.6 Opções de Inicialização da Replicação

Você deve utilizar a opção `server-id` no master e no slave para estabelecer uma ID de conexão única em cada servidor. Você deve escolher um valor único no intervalo de 1 a 2^32-1 para cada master e slave. Example: `server-id=3`

As opções que você pode utilizar no servidor master para controle do log binário estão todas descritas em Seção 4.10.4 [Log binário], Página 374.

A seguinte tabela descreve as opções que você pode utilizar nos servidores slaves. Você pode especificá-las na lina de comando ou no arquivo de opção.

NOTA: A replicação trata das seguintes opções de um modo especial:

- `--master-host`
- `--master-user`
- `--master-password`
- `--master-port`
- `--master-connect-retry`

Se não existir nenhum arquivo '`master.info`' quando o servidor slave inicia, ele usa valores especíificados no arquivo de opções ou na linha de comando. Isto irá ocorrer quando você iniciar o servidor como um slave de replicação pela primeira vez, ou você executar `RESET SLAVE` e desliga e reiniciar o servidor slave.

No entanto, se o arquivo '`master.info`' existe quando o servidor slave iniciar, ele usa o valor no arquivo e **IGNORA** qualquer valor especificado para aquelas opções no arquivo de opção ou na linha de comando.

Suponha que você especifique esta opção em seu arquivo '`my.cnf`':

```
[mysqld]
master-host=this_host
```

A primeira vez que você iniciar o servidor como um slave de replicação, ele irá ler e usar a opção do arquivo '`my.cnf`'. O servidor gravará então aquele valor no arquivo '`master.info`'.

A próxima vez que você iniciar o servidor, ele irá ler o valor da máquina master a partir do arquivo '`master.info`'. Se você modificar o arquivo '`my.cnf`' para especificar uma máquina master diferente, ele não terá efeito. Você deve usar `CHANGE MASTER TO`.

A partir do MySQL 4.1.1, as seguintes opções também é tratada de forma especial:

- `--master-ssl`
- `--master-ssl-ca`
- `--master-ssl-capath`
- `--master-ssl-cert`
- `--master-ssl-cipher`
- `--master-ssl-key`

O arquivo '`master.info`' inclui os valores correspondentes a essas opções. Adicionalmente, o formato do arquivo na versão 4.1.1 inclui na sua primeira linha o número de linhas no arquivo. Se você atualizar um servidor mais antigo para a versão 4.1.1, o '`master.info`' será atualizado para o novo formato automaticamente quando o novo servidor iniciar. (Se você substituir um MySQL 4.1.1 ou mais novo por uma versão mais antiga que a 4.1.1, você deve remover a primeira linha manualmente antes de iniciar o servidor mais antigo pela primeira vez.)

Como o servidor da precedência a uma arquivo '`master.info`' existente sobre as opções de inicialização acima descrito, você pode preferir usar as opções de inicialização para estes valores, e especifique-os usando a instrução `CHANGE MASTER TO`. See Seção 4.11.8.1 [`CHANGE MASTER TO`], Página 402.

Este exemplo mostra um uso mais extensivo das opções de inicialização para configurar um servidor slave:

```
[mysqld]
server-id=2
master-host=db-master.mycompany.com
master-port=3306
master-user=pertinax
master-password=freitag
master-connect-retry=60
report-host=db-slave.mycompany.com
```

The following list describes startup options for controlling replication:

`--log-slave-updates`

Diz ao slave para registrar as atualizações feitas pela thread da SQL do slave no log binário do slave. É desligado por padrão. É claro que ele exige que ele exige que o slave seja iniciado com o log binário habilitado (opção `--log-bin`). `--log-slave-updates` é usado quando você deseja colocar diversos servidores em cadeia. Por exemplo, você pode querer uma configuração como esta:

```
A -> B -> C
```

Isto é, A é o servidor master do slave B, e B é o servidor master do slave C. Para isto funcionar, onde B é tanto uma master quanto um slave, você deve iniciar B com a opção `--log-slave-updates`. A e B devem ser iniciados com o log binário habilitado.

`--log-warnings`
: Fazer slave exibir mais mensagens sobre o que está sendo feito. Por exemplo, ele avisará que ele obteve sucesso em reconectar depois de uma falha de conexão/rede, o avisrá sobre cada thread slave iniciada.

  Esta opção não está limitada apenas ao uso da replicação. Ela produz avisos através de um espectro de servidores ativos.

`--master-host=host`
: Especifica o nome de máquina ou endereço de IP do master para replicação. Se esta opção não for dada, a thread slave não será iniciada. O valor em '`master.info`' tem precedência se ele puder ser lido. Provavelmemte um nome nelhor para está opção seria algo do tipo `--bootstrap-master-host`, mas é muito tarde para alterá-la agora.

`--master-user=nome_usuário`
: O usuário da conta que a thread slave usará para autenticar ao conectar ao master. A conrta deve ter o privilégio `REPLICATION SLAVE` (Em versões anteriores a 4.0.2 ele devia ter o privilégio `FILE`). Se o usuário do master não for configurado, assume-se o usuário `teste`. O valor em '`master.info`' toma precedência se puder ser lida.

`--master-password=password`
: A senha da conta com a qual a thread slave autenticará quando conectar ao master. Se não definida, um senha vazia é considerada. O valor em '`master.info`' toma precedência se puder ser lido.

`--master-port=portnumber`
: A porta que o master está escutando. Se não definifa, a configuração de compilação do `MYSQL_PORT` é consierada. Se você não alterou as opções do `configure` , ela deve ser 3306. O valor em '`master.info`' toma precedência se ele puder ser lido.

`--master-connect-retry=seconds`
: O número de segundos que a thread slave espera antes de tentar se conectar ao master no caso do master ter caído ou a conexão for perdida. O padrão é 60. O valor em '`master.info`' toma precedência se puder ser lido.

`--master-info-file=filename`
: Especifica o nome a ser usado no arquivo que o slave grava a informação sobre o master. O nome padrão é `master.info` no diretório de dados.

`--master-ssl`
`--master-ssl-ca=file_name`
`--master-ssl-capath=directory_name`
`--master-ssl-cert=file_name`
`--master-ssl-cipher=cipher_list`
`--master-ssl-key=filename`
: Estas opções são usadas para configurar um conexão de replicação segura para o servidor master usando SSL. Os seus significados são os mesmos das opções correspondentes `--ssl`, `--ssl-ca`, `--ssl-capath`, `--ssl-cert`, `--ssl-cipher`, `--ssl-key` descritas em Seção 4.4.10.5 [SSL options], Página 274.

Estas opções estão operacionais a partir do MySQL 4.1.1.

`--max-relay-log-size=#`
: Para rotacionar o relay log automaticamente. Veja Seção 4.6.8.4 [`SHOW VARIABLES`], Página 309.

`--relay-log=filename`
: Para especificar a localização e nome que deve ser usado os relay logs. Você pode usá-lo para ter nomes de relay logs independentes do nome de máquina, ou se o seu relay log tend a ser grande (e você não que diminuir `max_relay_log_size`) e você precisa colocá-los em alguma área diferente do diretório de dados, ou se você quiser aumentar a velocidade balanceando as cargas entre os discos.

`--relay-log-index=filename`
: Para especificar a localização e nome que deve ser usado para arquivo de índice dos relay logs.

`--relay-log-info-file=filename`
: Para dar outro nome a '`relay-log.info`' e/ou colocá-lo em outro diretório, diferente do diretório de dados.

`--relay-log-purge=0|1`
: Disabilita/habilita a remoção automática dos relay logs assim que ele não são mais necessários. Esta é uma variável global que ode ser alterada dinâmicamente com `SET GLOBAL RELAY_LOG_PURGE=0|1`. o valor padrão é 1.

  Esta opção está disponível a partir do MySQL 4.1.1.

`--relay-log-space-limit=#`
: Para colocar um limite superior no tamanho total de todos os relay logs no slave (Um valor 0 significa "ilimitado"). Isto é útil se você tiver um disco rígido pequeno em. sua máquina slave. Quando o limite é alcançado, a thread de E/S fica em pausa (não lê o log binário do master) até que a thread de SQL tenha buscado e deletado alguns dos relay logs não utilizados. Note que este limite não é absoluto: existem casos onde a thread SQL precisa de mais eventos para poder deletar, neste caso a thread de E/S irá superar o limite até que a deleção seja possível. Se isto não for feito ela entra em deadlock (o que acontecia antes do MySQL 4.0.13). Os usuários não devem configurar `--relay-log-space-limit` para menos que duas vezes o valor de `--max-binlog-size` (ou `--max-binlog-size` se `--max-relay-log-size` for 0) porque neste caso há a chance de que quando a thread de E/S espera por espaço livre porque `--relay-log-space-limit` é excedido, a thread de SQL não tem relay log para apagar e assim não pode satisfazer a thread de E/S, forçando-a a ignorar temporariamente `--relay-log-space-limit`.

`--replicate-do-table=db_name.nome_tabela`
: Diz para thread slave restrigir a replicação a uma tabela específica. Para especificar mais de uma tabela, use a diretiva múltiplas vezes, uma para cada tabela. Isto funcionará para atualizações através de bancos de dados, em contraste com `--replicate-do-db`. Por favor, leia as notas que seguem esta lista de opções

`--replicate-ignore-table=db_name.nome_tabela`

Diz a thread slave para não replicar qualquer comando que atualiza a tabela especificada (mesmo se qualquer outra tabela puder ser atualizada pelo mesmo comando). Para especificar mais de uma tabela a ser ignorada, use a diretiva várias vezes, para cada tabela. Isto funcionará para atualizações através de bancos de dados, em contraste com `--replicate-ignore-db`. Por favor, leia as notas que seguem esta lista de opções

`--replicate-wild-do-table=db_name.nome_tabela`

Diz a thread slave para restringir a replicação a consultas onde qualquer das tabelas atualizadas correspondam a padrão de meta caracteres especificado. Para especificar mais de uma tabela, use a diretiva vária vezes, uma para cada tabela, Isto funciona para atualizações através de banco de dados. Por favor, leia as notas que seguem esta lista de opções

Exemplo: `--replicate-wild-do-table=foo%.bar%` replicará apenas atualizações que usam uma tabela em qualquer banco de dadis que comece com `foo` e cujos nomes de tabelas comecem com `bar`.

Note que se você fizer `--replicate-wild-do-table=foo%.%` então a regra será propagada para `CREATE DATABASE` e `DROP DATABASE`, ex.: estas duas instruções serão replicadas se o nome de banco de dados corresponder ao padrão do banco de dados (`'foo%'` aqui) (testa mágica é possível por `'%'` ser o padrão da tabela).

Caracteres curingas `_` e `%` escapados: se você quiser replicar, por exemplo, todas as tableas do banco de dados `my_own%db` (este é o nome exato do banco de dados), e não replicar tabelas do banco de dados `my1ownAABCdb`, você deve escapar o `_` e `%`: você deve usar algo como isto: `replicate-wild-do-table=my\_own\%db`. E se você estiver especificando esta opção para a linha de comando, dependendo do seu sistema, você precisará escapar o `\` (por exemplo, com uma shell `bash`, você precisaria digitar `--replicate-wild-do-table=my\\_own\\%db`).

`--replicate-wild-ignore-table=db_name.nome_tabela`

Diz a thread slave pra não replicar um consulta onde qualquer tabela corresponda ao padrão de meta caracteres dado. Para especificar mais de uma tabela, use a diretiva várias vezes, uma vez para cada tabela. Isto funcionará para atualizações através de banco de dados. Por favor, leia as notas que seguem esta lista de opções

Exemplo: `--replicate-wild-ignore-table=foo%.bar%` não atualizará tabelas no banco de dados que iniciar com `foo` e cujo os nomes de tabela iniciem com `bar`.

Note que se você fizer `--replicate-wild-ignore-table=foo%.%` então a regra será propagada para `CREATE DATABASE` e `DROP DATABASE`, ex. estas duas instruções não serão replciadas se o nome do banco de dados não corresponder ao padrão (`'foo%'` aqui) (esta mágica ocorre devido ao `'%'` como padrão da tabela).

Caracteres curingas `_` e `%` escapados: veja as anotações na descrição de `replicate-wild-do-table` logo acima.

`--replicate-do-db=nome_bd`

Diz ao slave para restringir a replicação a comandos onde o banco de dados atual (p.ex., aquele selecionado por `USE`) é `nome_bd`. Para especificar mais de uym banco de dadosm use a diretiva várias vezes, uma vez por tabela. Note que isto não replicará consultas entre bancos de dados tais como `UPDATE algum_bd.alguma_tabela SET foo='bar'` se for selecionado outro banco de dados ou nenhum banco de dados. Se você precisa que atualizações entre bancos de dados funcionem, certifique-se de que você tem o MySQL 3.23.28 ou posterior, e use `--replicate-wild-do-table=db_name.%`. Por favor, leia as notas que seguem esta lista de opções

Exemplo do que não funciona como você espera: se o slave é iniciado com `--replicate-do-db=sales`, e você faz `USE prices; UPDATE sales.january SET amount=amount+1000;`, esta consulta não será replicada.

Se você precisar que atualizações entre bancos de dados funcionem, use `--replicate-wild-do-table=db_name.%`.

A principal razão para este comportamento de apenas verificar o banco de dados atual é que é difícil para um comando sozinho saber se deve ser replicado ou não; por exemplo se você está usando comandos delete ou update multi-tabelas que continuam entre múltiplos bancos de dados. Também é muito mais rápido verificar apenas o banco de dados atual.

`--replicate-ignore-db=nome_bd`

Diz ao slave para não replicar qualquer comando onde o banco de dados atual (p.ex. o selecionado por `USE`) é `nome_bd`. Para especificar mais bancos de daods use a diretiva diversas vezes, uma para cada banco de dados. Você não deve utilizar esta diretiva se você está usando atualização através de tabelas e você não quer que estas atualizações sejam replicadas. Por favor, leia as notas que seguem esta lista de opções

Exemplo do que não funcionaria como esperado: se o slave é iniciado com `--replicate-ignore-db=sales`, e você faz `USE prices; UPDATE sales.january SET amount=amount+1000;`, esta consulta será replicada.

Se você precisar de atualizações entre banco de dados funcione, use `--replicate-wild-ignore-table=db_name.%`.

`--replicate-rewrite-db=de_nome->para_nome`

Diz ao slave para traduzir o banco de dados atual (p.ex. aquele selecionado por `USE`) para `para_nome` se ele era `de_nome` no master. Apenas instruções envolvendo a tabela podem ser afetadas (`CREATE DATABASE`, `DROP DATABASE` não poderão), e apenas se `de_nome` era o banco de dados atual no master. Isto não funcionará para atualizações entre banco de dados. Note que a translação é feita antes das regras de `--replicate-*` serem testadas.

Exemplo: `replicate-rewrite-db=master_db_name->slave_db_name`

`--report-host=host`

O Nome de máquina ou número IP do slave a ser relatado ao master durante o registro do slave. Aparecerá na saída de `SHOW SLAVE HOSTS`. Deixe indefinido se você não quiser que o slave se registre no master. Note que ele não é suficiente

para o master simplesmente ler o número IP do slave fora dos sockets uma vez que o slave se conecte. Devido ao `NAT` e outros assuntos de roteamento,a quele IP pode não ser válido para se conectar ao slave a partir do master ou outras máquinas.

Esta opção está disponível a partir do MySQL 4.0.0.

`--report-port=portnumber`
: Porta para conexão do slave relatado ao master durante o registro do slave. Defina-o apenas se o slave está escutando por uma porta diferente da padrão ou se você tiver um tunel especial do master ou outros clientes para o slave. Se não tiver certeza, deixe esta opção indefinida.

  Esta opção está disponível a partir do MySQL 4.0.0.

`--skip-slave-start`
: Diz ao servidor slave para não iniciar a thread slave na iicialização do servidor. O usuário pode iniciá-las mais tarde com `START SLAVE`.

`--slave_compressed_protocol=#`
: Se 1, usa compactação no protocolo cliente/servidor se tanto o slave quanto o mester suportá-la.

`--slave-load-tmpdir=filename`
: Esta opção é igual ao valor da variável `tmpdir` por padrão. Quando a thread SQL do slave replica um comando `LOAD DATA INFILE`, ele extrai os arquivos a serem carregados do relay logs em arquivos temporários, e então os carrega dentro da tabela. Se o arquivo carregado no master era enorme, os arquivos temporários no slave também serão enormes; embora você possa desejar que o slave coloque o arquivo temporário em algum disco grande diferente de `tmpdir`, usando esta opção. Nestes caso, você também pode usar a opção `--relay-log`, já que os relay logs serão grandes também. `--slave-load-tmpdir` deve apontar para o sistema de arquivo baseado em disco; não em um baseado em memória. Como o slave precisa de arquivos temporários usados para replicar `LOAD DATA INFILE`) para sobreviver a uma reinicialização da máquina.

`--slave-net-timeout=#`
: Número de segundos a esperer por mais dados do master antes de abortar a leitura, considerando o quebra de conexão e as tentativas de reconectar. A primeira vez ocorre imediatamente depois do tempo limite. O intervalo entre tentativas é controlado pela opção `--master-connect-retry`.

`--slave-skip-errors= [err_code1,err_code2,... | all]`
: Diz ao a thread SQL do slave para continuar a replicação quando uma consulta retornar um erro de uma lista fornecida. Normalmente, a replicação irá parar ao encontrar um erro, dando ao usuário a chance de resolver a inconsistêncian nos dados manualmente. Não use esta opção a menos que você saiba exetamente o motivo dos erros. Se não houver erros em sua configuração da replicação e programas clientes, e não houver erros no MySQL, você nunca deve ter uma replicação abortada com erro. O uso indiscriminado desta opção resultará em slaves fora de sincronia com o master e você não terá idéia de como o problema aconteceu.

Para códigos de erros, você deve usar o número fornecido pela mensagem de erron no seu log de erros do slave e na saída de `SHOW SLAVE STATUS`. Uma lista completa de mensagens de erro podem ser encontradas na distribuição fonte em '`Docs/mysqld_error.txt`'. Os códigos de erros do servidor também são listados em Seção 13.1 [Error-returns], Página 889.

Você também pode (mas não deve) usar um valor não recomendado de `all` o que irá ignorar todas as mensagens de erro e continua em frente indiferentemente. Não é preciso dizer, que se você usar isto, não podemos garantir a integridade dos seus dados. Por favor, não reclame se seus dados no slave não estiver nem próximo daqueles em seu master neste caso — você foi avisado.

Exemplos:

```
--slave-skip-errors=1062,1053
--slave-skip-errors=all
```

Algumas destas opções, como todas as opções `--replicate-*`, só podem ser definidas na inicialização do servidor slave, e não com ele ligado. Planejamos corrigir isto.

Aqui está a ordem de avaliação das regras `--replicate-*`, para decidir se a consulta será executada pelo slave ou ignorada por ele:

1. Existe alguma regra `--replicate-do-db` ou `--replicate-ignore-db`?
   - Sim: teste-as como para `--binlog-do-db` e `--binlog-ignore-db` (veja Seção 4.10.4 [Binary log], Página 374). Qual é o resultado do teste?
     - ignore a consulta: ignore-a e saia.
     - execute a consulta: não execute-a imediatamente, adie a decisão, vá para o passo abaixo.
   - Não: vá para o passo abaixo.
2. Existe alguma regra `--replicate-*-table`?
   - Não: execute a consulta e saia.
   - Sim: vá para o passo abaixo. Apenas tabela que serão atualizadas serão comparadas às regras (`INSERT INTO sales SELECT * from prices`: apenas `sales` será comparada às regras). Se várias tabelas forem ser atualizadas (instruções multi-tabelas) a primeira a corresponder a regra (com "do" ou "ignore") vence (isto é, a primeira tabela é comparada a regra. se nenhuma decisão pode ser tomada a segunda tabela é compara às regras, etc).
3. Existe alguma regra `--replicate-do-table`?
   - Sim: o tabela encaixa em alguma delas?
     - Sim: execute a consulta e saia.
     - Não: vá para o passo abaixo.
   - Não: vá para o passo abaixo.
4. Existe alguma regra `--replicate-ignore-table`?
   - Sim: a tabela encaixa em alguma delas?
     - Sim: ignore a consulta e saia.
     - Não: vá para o passo abaixo.
   - Não: vá para o passo abaixo.

5. Existe alguma regra `--replicate-wild-do-table`?
   - Sim: a tabela se encaixa em qualquer uma delas?
     - Sim: execute a consulta e saia.
     - Não: vá para o passo abaixo.
   - Não: vá para o passo abaixo.
6. Existe alguma regra `--replicate-wild-ignore-table`?
   - Sim: a tabela se encaixa em qualquer uma delas?
     - Sim: ignore a consulta e saia.
     - Não: vá para o passo abaixo.
   - Não: vá para o passo abaixo.
7. Nenhuma regra `--replicate-*-table` foi correspondida. Existe outra tabela para se testar com estas regras?
   - Sim: loop.
   - Não: testamos todas as tabelas a serem atualizadas, nenhuma regra foi obedecida. Existem regras `--replicate-do-table` ou `--replicate-wild-do-table`?
     - Sim: ignore a consulta e saia.
     - Não: execute a consulta e saia.

4.11.7 Instruções SQL para Controle do Servidor Master

0 replicação pode ser controlada por meio da interface SQL. Esta seção discute instruções para gerenciamento dos servidores masters de replicação. Seção 4.11.8 [Replication Slave SQL], Página 402 discute instruções para gerenciamento dos servidores slaves.

4.11.7.1 PURGE MASTER LOGS

```
PURGE {MASTER|BINARY} LOGS TO 'log_name'

PURGE {MASTER|BINARY} LOGS BEFORE 'date'
```

Deleta todos os logs binários que estão listados no índice de log anteriores ao log ou data especificado. O log também remove da lista gravada no índice de log, e assim o log dado se torna o primeiro.

Exemplo:

```
PURGE MASTER LOGS TO 'mysql-bin.010';
PURGE MASTER LOGS BEFORE '2003-04-02 22:46:26';
```

A variante `BEFORE` está disponível no MySQL 4.1; este argumento de data pode estar no formato `'YYYY-MM-DD hh:mm:ss'`. `MASTER` e `BINARY` são sinônimos, embora `BINARY` possa ser usado apenas a partir do MySQL 4.1.1.

Se você tiver um slave ativo que está atualmente lendo um dos logs que você stá tentando deletar, este comando não faz nada e falha com um erro. No entanto, se você tiver um slave ativo e apagar um dos logs que ele quiser ler, o slave não poderá replicar uma vez que ele esteja ativo. O comando é seguro para de se executar enquanto os sslaves estiverem replicando. Você não precisa de pará-los.

Você deve primeiro verificar todos os slaves com `SHOW SLAVE STATUS` para ver qual log eles estão lendo, e então você deve fazer uma lista dos logs no master com `SHOW MASTER LOGS`, encontrar o log mais novo entre todos os slaves (se todos os slaves estão atualizados, ele será o último log da lista), tirar backup de todos os logs que você está prestes a deletar (opcional) e deletar até o log alvo.

4.11.7.2 RESET MASTER

```
RESET MASTER
```

Deleta todos os logs binários listado no arquivo de índice, zerando o arquivo de índice do log binário.

Esta instrução rea chamada `FLUSH MASTER` antes do MySQL 3.23.26.

4.11.7.3 SET SQL_LOG_BIN

```
SET SQL_LOG_BIN = {0|1}
```

Disabilita ou habilita o log binário para a conexão do usuário (`SQL_LOG_BIN` é uma variável de sessão) se o cliente conecta usando uma conta que tem o privilégio `SUPER`. A instrução é ignorada se o cliente não possui este privilégio.

4.11.7.4 SHOW BINLOG EVENTS

```
SHOW BINLOG EVENTS [ IN 'log_name' ] [ FROM pos ] [ LIMIT [offset,] row_count ]
```

Mostra o evento no log binário. Se você não especificar `'log_name'`, o primeiro log binário será exibido.

Esta instrução está disponível a partir do MySQL 4.0.

4.11.7.5 SHOW MASTER STATUS

```
SHOW MASTER STATUS
```

Fornece a informação de status no log binário do master.

4.11.7.6 SHOW MASTER LOGS

```
SHOW MASTER LOGS
```

Lista o log binário no master. Você deve usar este comando antes de `PURGE MASTER LOGS` para descobrir até onde você deve ir.

4.11.7.7 SHOW SLAVE HOSTS

```
SHOW SLAVE HOSTS
```

Mostra uma lista de slaves atualmente registrados com o master. Note que slaves não iniciados com a opção `--report-host=slave_name` não estarão visíveis nesta lista.

4.11.8 Instruções SQL para Controle do Servidor Slave

A replicação pode ser controlada por meio da interface SQL. Esta seção discute instruções para gerenciamento dos servidores slaves de replicação. Seção 4.11.7 [Replication Master SQL], Página 400 discute instruções para gerenciamento dos servidores master.

4.11.8.1 CHANGE MASTER TO

```
CHANGE MASTER TO master_def [, master_def] ...

master_def =
      MASTER_HOST = 'host_name'
    | MASTER_USER = 'user_name'
    | MASTER_PASSWORD = 'password'
    | MASTER_PORT = port_num
    | MASTER_CONNECT_RETRY = count
    | MASTER_LOG_FILE = 'master_log_name'
    | MASTER_LOG_POS = master_log_pos
    | RELAY_LOG_FILE = 'relay_log_name'
    | RELAY_LOG_POS = relay_log_pos
    | MASTER_SSL = {0|1}
    | MASTER_SSL_CA = 'ca_file_name'
    | MASTER_SSL_CAPATH = 'ca_directory_name'
    | MASTER_SSL_CERT = 'cert_file_name'
    | MASTER_SSL_KEY = 'key_file_name'
    | MASTER_SSL_CIPHER = 'cipher_list'
```

Altera os parâmetros que o servidor slave usa para conectar e comunicar com o servidor master. Os valores possíveis para o valor `master_def` estão mostrados acima.

As opções do relay log (`RELAY_LOG_FILE` e `RELAY_LOG_POS`) estão disponíveis a partir do MySQL 4.0.

As opções SSL (`MASTER_SSL`, `MASTER_SSL_CA`, `MASTER_SSL_CAPATH`, `MASTER_SSL_CERT`, `MASTER_SSL_KEY`, e `MASTER_SSL_CIPHER`) estão disponíveis a partir do MySQL 4.1.1. Você pode alterar estas opções mesmo nos slaves que são compilados sem suporte a SSL. Eles serão salvos no arquivo '`master.info`' mas ignorados até que você use um servidor que tenha suporte a SSL habilitado.

Por exemplo:

```
mysql> CHANGE MASTER TO
    ->     MASTER_HOST='master2.mycompany.com',
    ->     MASTER_USER='replication',
    ->     MASTER_PASSWORD='bigs3cret',
    ->     MASTER_PORT=3306,
    ->     MASTER_LOG_FILE='master2-bin.001',
    ->     MASTER_LOG_POS=4,
    ->     MASTER_CONNECT_RETRY=10;
mysql> CHANGE MASTER TO
```

```
    ->     RELAY_LOG_FILE='slave-relay-bin.006',
    ->     RELAY_LOG_POS=4025;
```

MASTER_USER, MASTER_PASSWORD, MASTER_SSL, MASTER_SSL_CA, MASTER_SSL_CAPATH, MASTER_SSL_CERT, MASTER_SSL_KEY, and MASTER_SSL_CIPHER are information for the slave to be able to connect to its master. If you don't specify some of these informations, the non-specified informations will keep their old value. For example, if the password to connect to your MySQL master has changed, you just need to issue

```
mysql> STOP SLAVE; -- if replication was running
mysql> CHANGE MASTER TO MASTER_PASSWORD='new3cret';
mysql> START SLAVE; -- if you want to restart replication
```

to tell the slave about the new password; no need to specify the information which did not change (host, port, user etc).

MASTER_HOST, MASTER_PORT are the hostname or IP adress of the master host, and its TCP port. Note that if MASTER_HOST is equal to `localhost`, then, like in other parts of MySQL, the port may be ignored (if Unix sockets can be used for example).

Se você especificar MASTER_HOST ou MASTER_PORT, o slave assumirá que o mestre é diferente do anterior (mesmo se você especificar um valor de nost ou porta iguais ao do valor atual.) Neste caso Assim, os valores antigos do nome e posição do log binário do mestre não são mais aplicáveis, assim se você não especificar MASTER_LOG_FILE e MASTER_LOG_POS no comando, MASTER_LOG_FILE='' e MASTER_LOG_POS=4 são silenciosamente adicionados a ele.

MASTER_LOG_FILE e MASTER_LOG_POS são as coordenadas das quais a thread de E/S do slave começara a ler do master na próxima vez em que ele for iniciado. If you specify any of them, you can't specify RELAY_LOG_FILE or RELAY_LOG_POS. If none of MASTER_LOG_FILE and MASTER_LOG_POS was specified, then the last coordinates of the **slave SQL thread** before CHANGE MASTER was issued, are used. This ensures that replication has no discontinuity, even if the slave SQL thread was late compared to the slave I/O thread, when you just want to change, say, the password to use. This safe behaviour was introduced starting from MySQL 4.0.17 and 4.1.1. (Before these versions, the used coordinates were the last coordinates of the slave I/O thread before CHANGE MASTER was issued, which caused the SQL thread to sometimes lose some events from the master, thus breaking replication.)

CHANGE MASTER TO **deleta todos os relay logs** (e inicia um novo), a menos que você especifique RELAY_LOG_FILE ou RELAY_LOG_POS (neste caso os relay logs serão mantidos; desde o MySQL 4.1.1 a variável global RELAY_LOG_PURGE será definida com zero sem aviso prévio). CHANGE MASTER TO atualiza '`master.info`' e '`relay-log.info`'.

CHANGE MASTER é util para configurar um slave quando você tem a cópia do master e gravou o registro e offset no master que corresponde a cópia tirada. Você pode executar CHANGE MASTER TO MASTER_LOG_FILE='log_name_on_master', MASTER_LOG_POS=log_offset_on_master no slave depois de restaurar a cópia.

O primeiro exemplo acima (CHANGE MASTER TO MASTER_HOST='master2.mycompany.com' etc) altera as coordenadas do master e do seu log binário. Isto é quando você deseja que o slave replique o master. O segundo exemplo, usado com menos frequência, é quando o slave possui relay logs que, por alguma razão, você deseja que o slave execute novamente; para fazer isto o master não precisa estar alcançavel, você só precisa fazer CHANGE MASTER TO e iniciar a thread de SQL (START SLAVE SQL_THREAD). Você pode usar isto mesmo fora da

consiguração de replicação, em um servidor standalone, slave-de-ninguém, para recuperação depois de uma falha.

Suponha que o seu servidor tenha falhado e você tenha restaurado um backup. Você deseja reexecutar o próprio log binário do servidor (não os relay logs, mas logs binários regulares), supostamente chamado '`myhost-bin.*`'. Primeiro faça uma cópia destes logs binários em alguns lugares seguros, no caso de você não seguir exatamente o procedimento abaixo e acidentalmente apagar os logs binários de servidor. Se você estiver usando o MySQL 4.1.1 ou mais novos, defina `SET GLOBAL RELAY_LOG_PURGE=0` para segurança adicional. Então inicie o servidor sem `log-bin`, com um novo ID do servidor (diferente do anterior), com `relay-log=myhost-bin` (para fazer o servidor acreditar que estes logs binários regulares são relay logs) e `skip-slave-start`, então execute estas instruções:

```
mysql> CHANGE MASTER TO
    ->     RELAY_LOG_FILE='myhost-bin.153',
    ->     RELAY_LOG_POS=410,
    ->     MASTER_HOST='some_dummy_string';
mysql> START SLAVE SQL_THREAD;
```

Então o servidor irá ler e executar seus próprios logs binários, e assim conseguindo a recuperação de falhas. Uma vez que a recuperação está finalizada, execute `STOP SLAVE`, desligue o servidor, delete '`master.info`' e '`relay-log.info`', e reinicie o servidor com suas opções originais. No momento, especificar `MASTER_HOST` (mesmo com um valor modelo) é compulsório para fazer o servidor pensar que ele é um slave, e dar ao servidor um novo ID, diferente do anterior é compulsório senão o servidor verá os eventos com seus IDs e pensará que ele está em uma configuração de replicação circular e ignora os eventos, o que é indesejado. No futuro planejamos adicionar opções para lidar com estas pequenas restrições.

4.11.8.2 LOAD DATA FROM MASTER

```
LOAD DATA FROM MASTER
```

Tira uma cópia do master para o slave. Atualiza os valores de `MASTER_LOG_FILE` e `MASTER_LOG_POS` assim o slave será iniciado replicando da posição correta. Respeitará a regras de exclusão de tabelas e bancos de dados especificadas com as opções `replicate-*`.

O uso desta instrução está sujeito ao seguinte:

- Funciona apenas com tabelas `MyISAM`.
- Ele adquire um lock de leitura global no master enquanto tira um instantâneo, que evita atualizações no master durante esta operação.

No futuro está planejado fazê-lo funcionar com tabelas `InnoDB` e remover a necessidade de lock deleitura global usando o recurso de backup online sem bloqueio.

Se você estiver carregando tabelas grandes, você pode aumentar os valores de `net_read_timeout` e `net_write_timeout` no mestre e no slave. Veja Seção 4.6.8.4 [SHOW VARIABLES], Página 309.

Note que `LOAD DATA FROM MASTER` **NÂO** copia nenhuma tabela do banco de dados `mysql`. Isto é para tornar facil de se ter diferentes usuários e privilégios no master e no slave.

Esta instrução exige que o usuário de replicação usado para se conectar ao master tenha privilégios `RELOAD` e `SUPER` no master, privilégios `SELECT` em todas as tabelas do master

que você queira carregar. Todas as tabelas do master nas quais os usuários não tenham privilégio SELETC serão ignoradas pelo LOAD DATA FROM MASTER; isto ocorre porque o master irá esconde-los do usuário: LOAD DATA FROM MASTER chama SHOW DATABASES para saber qual banco de dados do master carregar, mas SHOW DATABASES retorna apenas o banco de dados nos quais o usuário tem algum privilégio. Veja Seção 4.6.8.1 [Show database info], Página 303. No lado do slave, o usuário que executa LOAD DATA FROM MASTER deve ter permissão para apagar e criar o banco de dados e tabelas envolvidos.

4.11.8.3 LOAD TABLE tbl_name FROM MASTER

```
LOAD TABLE tbl_name FROM MASTER
```

Faz o download de uma cópia da tabela do master para o slave. Esta instrução é implementada principalmente para depuração de LOAD DATA FROM MASTER. Exige que o usuário de replicação que é usado para conectar ao master tenha privilégios RELOAD e SUPER no master, e SELECT na tabela do master que será carregada. No lado do slave, o usuário que envia LOAD TABLE FROM MASTER deve ter permissão para apagar e criar a tabela. Leia as anotações sobre tempo limite nadescrição de LOAD DATA FROM MASTER abaixo, elas se aplicam aqui também. Por favor, leia também as limitações de LOAD DATA FROM MASTER acima, elas também se aplicam (por exemplo, LOAD TABLE FROM MASTER só funciona com tabelas MyISAM).

4.11.8.4 MASTER_POS_WAIT()

```
SELECT MASTER_POS_WAIT('master_log_file', master_log_pos)
```

Esta é uma função, não um comando. É usada para assegurar que o slave tenha alcançado (lido e executado) uma dada posição no log binário do master. Veja Seção 6.3.6.2 [Miscellaneous functions], Página 547 para uma descrição completa.

4.11.8.5 RESET SLAVE

```
RESET SLAVE
```

Faz o slave esquecer a sua posição de replicação no log binário do master. Esta instrução é usada para uma inicialização limpa: ela deleta os arquivos '`master.info`' e '`relay-log.info`', todos os relay logs e inicia um novo relay log. **Nota:** Todos os relay logs são deletados, mesmo se não forem totalmente executados pela threads SQL do slave. (Esta é uma condição que deveria existir em um slave de replicação altamente carregado, ou se você enviasse uma instrução STOP SLAVE.) As informações de conexão armazenadas no arquivo '`master.info`' são imediatamente recarregadas com os valores especificados nas opções de inicializacão, se forem especificadas. Estas informações incluem valores como máquina master, porta do master, usuário do master e senha do master. Se a thread SQL do slave estava no meio de uma replicação de tabelas temposrárias quando ela foi parada, e RESET SLAVE é excutado, estas tabelas temporárias replicadas são deletadas no slave.

Esta instrução era chamada FLUSH SLAVE antes do MySQL 3.23.26.

4.11.8.6 SET GLOBAL SQL_SLAVE_SKIP_COUNTER

```
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = n
```

Salta os próximos `n` eventos do master. Útil para recuperação de paradas da replicação causada por um erro.

Esta instrução só é válida quando a thread slave não está em execução, em caso contrário, retorna um erro.

Antes do MySQL 4.0, omite a palavra chave `GLOBAL` da instrução.

4.11.8.7 SHOW SLAVE STATUS

```
SHOW SLAVE STATUS
```

Fornece a informação de status nos parâmetros essenciais da thread do slave. Se você utilizar esta instrução usando no cliente `mysql`, você pode usar o terminador `\G` em vez de um ponto e vírgula no fim, para conseguir um layout vertical mais legível:

```
mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
       Slave_IO_State: Waiting for master to send event
          Master_Host: localhost
          Master_User: root
          Master_Port: 3306
        Connect_Retry: 3
      Master_Log_File: gbichot-bin.005
  Read_Master_Log_Pos: 79
       Relay_Log_File: gbichot-relay-bin.005
        Relay_Log_Pos: 548
Relay_Master_Log_File: gbichot-bin.005
     Slave_IO_Running: Yes
    Slave_SQL_Running: Yes
      Replicate_Do_DB:
  Replicate_Ignore_DB:
           Last_Errno: 0
           Last_Error:
         Skip_Counter: 0
  Exec_Master_Log_Pos: 79
      Relay_Log_Space: 552
      Until_Condition: None
       Until_Log_File:
       Until_Log_Pos: 0
   Master_SSL_Allowed: No
   Master_SSL_CA_File:
   Master_SSL_CA_Path:
      Master_SSL_Cert:
    Master_SSL_Cipher:
       Master_SSL_Key:
Seconds_Behind_Master: 8
```

Dependendp da sua versão do MySQL, você pode não ver todos os campos como aqui mostrado. Alguns campos estão presentes apenas a partir do MySQL 4.1.1.

Os campos mostrados por `SHOW SLAVE STATUS` tem o seguinte significado:

`Slave_IO_State`
: Uma cópia da coluna `State` da saída de `SHOW PROCESSLIST` para a thread de E/S do slave; lhe dirá se está thread está tentando se conectar ao master, esperando por eventos do master, reconectando ao master, etc. Os estados possíveis estão listados em Seção 4.11.2 [Replication Implementation], Página 379. Olhar está coluna é necessário porque, por exemplo, a thread pode estar em execução mas não tem sucesso ao tentar se conectar ao master: apenas esta coluna lhe deixará ciente do problema de conexão. Por outro lado, o estado da thread SQL não é copiada, porque as coisas são mais simples para esta thread: se ela estiver em execução, não haverá problema; se não, você encontrará o erro na coluna `Last_Error` (descrita abaixo).

  Este campo está presente a partir do MySQL 4.1.1.

`Master_Host`
: A máquina master atual.

`Master_User`
: O usuário usado para conectar ao master.

`Master_Port`
: A porta atual do master.

`Connect_Retry`
: O valor atual de `master-connect-retry`.

`Master_Log_File`
: O nome do arquivo de log binário do master no qual a thread de E/S está lendo atualmente.

`Read_Master_Log_Pos`
: A posição até a qual a thread de E/S leu no log binário do master.

`Relay_Log_File`
: O nome do arquivo de relay log na qual a thread SQL está lendo e executando atualmente.

`Relay_Log_Pos`
: A posição até a qual a thread de SQL leu e executou neste relay log.

`Relay_Master_Log_File`
: O nome do arquivo de log binário do master no qual contém o último evento executado pela thread de SQL.

`Slave_IO_Running`
: Diz se a thread de E/S foi iniciada ou não.

`Slave_SQL_Running`
: Diz se a thread de SQL está iniciada ou não.

`Replicate_Do_DB, Replicate_Ignore_DB`
: A lista de banco de dados que foi especificado com as opções `--replicate-do-db` e `--replicate-ignore-db`.

`Replicate_Do_Table`, `Replicate_Ignore_Table`, `Replicate_Wild_Do_Table`, `Replicate_Wild_Ignore_Table`

As tabelas que foram especificadas com as opções `--replicate-do-table`, `--replicate-ignore-table`, `--replicate-wild-do-table`, e `--replicate-wild-ignore_table`.

Estes campos estão presentes a partir do MySQL 4.1.1.

`Last_Errno`

O número de erro retornado pela consulta executada mais recentemente. Um valor 0 significa "sem erro".

`Last_Error`

A mensagem de erro retonada pela consulta executada mais recentemente. Por exemplo:

```
Last_Errno: 1051
Last_Error: error 'Unknown table 'z'' on query 'drop table z'
```

A mensagem indica que a tabela `z` existia no mestre e foi apagada lá, mas ela não existe no slave, assim `DROP TABLE` falhou no servidor. (Isto pode ocorrer se o usuário esqueceu de copiá-la no slave ao configurá-lo).

A string vazia significa "sem erro". Se o valor `Last_Error` não for vazio, ele também apareceria como uma mensagem no log de erro do slave. Por exemplo:

`Skip_Counter`

O último valor usado por `SQL_SLAVE_SKIP_COUNTER`.

`Exec_Master_Log_Pos`

A posição no log binário do master (`Relay_Master_Log_File`) do último evento executado pela thread de SQL. ((`Relay_Master_Log_File`,`Exec_Master_Log_Pos`) no log binário do master corresponde a (`Relay_Log_File`, `Relay_Log_Pos`) no relay log).

`Relay_Log_Space`

O tamanho total de todos os relay logs existentes.

`Until_Condition`, `Until_Log_File`, `Until_Log_pos`

O valor especificado na cláusula `UNTIL` da instrução `START SLAVE`.

`Until_Condition` possui estes valores estes valorer:

- `None` se nenhuma cláusula `UNTIL` foi especificada
- `Master` se o slave estiver lendo até uma dada posição no log binário do master.
- `Relay` se o slave estiver lendo até uma dada posição em seus relay logs

`Until_Log_File` e `Until_Log_Pos` indicam o nome do arquivo de log e e posição que define o ponto no qual a thread SQL irá parar a execução.

Estes campos estão presentes a partir do MySQL 4.1.1.

`Master_SSL_Allowed`, `Master_SSL_CA_File`, `Master_SSL_CA_Path`, `Master_SSL_Cert`, `Master_SSL_Cipher`, `Master_SSL_Key`

Estes campos mostram os parâmetros SSL usado pelo slave para se conectar os master, se existirem.

`Master_SSL_Allowed` possui estes valores:

- `Yes` se uma conexão SSL ao master é permitida
- `No` se uma conexão SSL ao master não é permitida
- `Ignored` se uma conexão SSL permitida pelo servidor slave não tem suporte a SSL habilitado.

Os valores dos outros campos correspodem ao valor das opções `--master-ca`, `--master-capath`, `--master-cert`, `--master-cipher`, e `--master-key`.

Estes campos estão presentes a partir do MySQL 4.1.1.

`Seconds_Behind_Master`

O número de segundos passados desde o último evento do master executado pela thread salve SQL. Será `NULL` quando nenhum evento foi executado ainda, ou depois de `CHANGE MASTER` e `RESET SLAVE`. Esta coluna pode ser usada para saber "quão atrasado está o seu slave". Funcionará mesmo se o seu master e slave não tiverem clocks idênticos.

Estes campos estão presentes a partir do MySQL 4.1.1.

4.11.8.8 START SLAVE

```
START SLAVE [thread_name [, thread_name] ... ]
START SLAVE [SQL_THREAD] UNTIL
    MASTER_LOG_FILE = 'log_name', MASTER_LOG_POS = log_pos
START SLAVE [SQL_THREAD] UNTIL
    RELAY_LOG_FILE = 'log_name', RELAY_LOG_POS = log_pos

thread_name = IO_THREAD | SQL_THREAD
```

`START SLAVE` sem nenhuma opção inicia ambas as threads slaves. A thread de E/S lêem as consultas do servidor master e as armazena no relay logs. A thread de SQL le o relay log e executa a consulta. Note que se `START SLAVE` obter sucesso no inicialização da thread slave ela retornará sem qualquer erro. Mas mesmo neste caso pode ser que a thread slave iniciou e parou mais tarde (por que elas não conseguiram se conectar ao master ou leram o seu log binário ou qualquer outro problema). `START SLAVE` não lhe avisará sobre insto. Você deverá verifica seu arquivo log de erro do slave por mensagens de erro gerada pela thread slave, ou verificar que eles estão rodando bem com `SHOW SLAVE STATUS`.

A partir do MySQL 4.0.2, você pode adicionar as opções `IO_THREAD` ou `SQL_THREAD` à instrução a ser chamada quando a thread iniciar.

A partir do MySQL 4.1.1, uma cáusula `UNTIL` pode ser adicionada para especificar que o slave deve iniciar até que a thread de SQL alcance um determinado ponto no log binário dp master ou no relay log do slave. Quando a thread SQL alcança este ponto, ela para. Se a opção `SQL_THREAD` é especificada na instrução, ela inicia apenas a thread de SQL. Senão, ela inicia ambas as threads slaves. Se a thread SQL já estiver em execução, a claúsula `UNTIL` é ignorada e um aviso é enviado.

Com uma cláusula `UNTIL`, você deve especificar tanto uma nome de arquivo de log quanto uma posição. Não misture opções do master e do relay logs.

Qualquer condição `UNTIL` é restaurada por uma instrução `STOP SLAVE` subsequente, ou uma instrução `START SLAVE` que não incluir a cláusula `UNTIL`, ou um servidor reinicie.

A cláusula `UNTIL` pode ser útil para depurar a replicação, ou para fazer com que a replicação proceda até um pouco antes do ponto que você deseja evitar que o slave replique uma instrução. Por exemplo, se uma instrução `DROP TABLE` foi executada no master, você pode usar `UNTIL` para dizer ao slave para executar até aquele ponto, mas não depois. Para encontrar qual é o evento, use `mysqlbinlog` com o log do master ou o relay logs, ou usando uma instrução `SHOW BINLOG EVENTS`.

Se você estiver usando `UNTIL` para ter o processo slave replicando consultas nas seções, é recomendado que você inicie o slave com a opção `--skip-slave-start` para evitar que a thread de SQL execute quando o slave iniciar. É provavelmente melhor usar esta opção em um arquivo de opção em vez de usá-la na linha de comando, assim uma reinicialização inesperada do servidor não faz com que isso seja esquecido.

A instrução `SHOW SLAVE STATUS` inclui campos na saída que mostram o valor atual da condição `UNTIL`.

Este comando é chamado `SLAVE START` antes do MySQL 4.0.5. No momento, `SLAVE START` ainda é aceito para compatibilidade com versões anteriores, mas está obsoleto.

4.11.8.9 STOP SLAVE

```
STOP SLAVE [thread_name [, thread_name] ... ]

thread_name = IO_THREAD | SQL_THREAD
```

Para a thread slave. Como o `START SLAVE`, esta instrução pode ser usada com as opções `IO_THREAD` e `SQL_THREAD` para chamar a thread ou threads que irão parar.

Este comando é chamado `SLAVE STOP` antes do MySQL 4.0.5. No momento, `SLAVE STOP` ainda é aceito para compatibilidade com versões anteriores, mas está obsoleto.

4.11.9 FAQ da Replicação

P: Como eu configuro um slave se o master já estiver em execução e eu não quiser pará-lo?

R: Existem diversas opções. Se você tirou um backup do master em alguns pontos e gravou o nome e offset do log binário (da saída do `SHOW MASTER STATUS`) correspondente à cópia, faça o seguinte:

1. Esteja certo de que o slave possuí um ID server único.
2. Execute as seguintes instruções no slave, preenchendo os valores apropriados para cada parâmetro:

   ```
   mysql> CHANGE MASTER TO
       ->     MASTER_HOST='master_host-name',
       ->     MASTER_USER='master_user_name',
       ->     MASTER_PASSWORD='master_pass',
       ->     MASTER_LOG_FILE='recorded_log_name',
       ->     MASTER_LOG_POS=recorded_log_pos;
   ```
3. Execute `START SLAVE` no slave.

Se você já não tiver um backup do master, aqui está um modo rápido de fazê-lo de forma consistente:

1. `FLUSH TABLES WITH READ LOCK`
2. `gtar zcf /tmp/backup.tar.gz /var/lib/mysql` (ou uma variação disto)
3. `SHOW MASTER STATUS` - esteja certo de gravar a saída - você precisará dela mais tarde
4. `UNLOCK TABLES`

Uma alternativa é tirar um dump do SQL do master em vez de uma cópia binária como acima; para isto você podee usar `mysqldump --master-data` em seu master e posteriormente executar este dump SQL em seu slave. Isto é, no entanto, mais lento que fazer uma cópia binária.

Não importa qual dos dois métodos você usa, mais tarde siga as instruções para o caso em que você tem uma cópia e gravou o nome e offset dos logs Você pode usar a mesma cópia para configurar diversos slaves. Uma vez que os logs binários do master estão intáctos, você pode esperar por dias ou meses para configurar um slave uma vez que você possui a cópia do master. Em teoria a lacuna de espera pode ser infinita. As duas limitações práticas é o espaço em disco do master sendo preenchido com logs antigos e a quantidade de tempo que o slave gastará para buscá-los.

Você também pode usar `LOAD DATA FROM MASTER`. Este é um comando conveniente que tira uma cópia, a restaurá no slave e ajustar o nome e offset do log no slave, todos de uma vez. No futuro, `LOAD DATA FROM MASTER` será o modo recomendado de configurar um slave. Esteja avisado, no entanto, que o lock de leitura pode ser mantido por um longo tempo se você usar este comando. Ele ainda não está implementado de forma tão eficiente quanto gostariamos. Se você tiver tabelas grandes, o método preferível neste momento ainda é com uma cópia `tar` local depois de executar `FLUSH TABLES WITH READ LOCK`.

P: O slave precisa estar conectado ao master o tempo todo?

R: Não, ele não precisa. O slave pode ser desligado ou permanecer desconectado por horas ou mesmo dias, então reconecta e busca as atualizações. Por exemplo, você pode usar uma relação master/slave sobre uma conexão dial-up que está ligada esporádicamente apenas por um curto período de tempo. A implicação disto é que a uma hora dada qualquer não temos garantias de que o slave está sincronizado com o master a menos que você tire algumas medidas especiais. No futuro, teremos a opção de bloquear o master até que pelo menos um slave esteja sincronizado.

P: Como posso saber se um slave está atrasado comparado ao master? Em outra palavras, como eu sei que o dado da última consulta replicada pelo escravo?

R: Se o slave for 4.1.1 ou mais novo, leia a coluna `Seconds_Behind_Master` de `SHOW SLAVE STATUS`. Para versão mais antigas o seguinte se aplica. Isto só é possível se a thread salve de SQL existir (p.ex. se ele for exibida em `SHOW PROCESSLIST`, veja Seção 4.11.3 [Replication Implementation Details], Página 381) (no MySQL 3.23: se a thread slave existir, p.ex. e mostrada em `SHOW PROCESSLIST`), e se ela executou pelo menos um evento do master. Realmente, quando a thread slave de SQL executa um evento lido do master, esta thread modifica seu próprio tempo do timestamp do evento (é por isto que `TIMESTAMP` é bem replicado). Assim é indicado na coluna `Time` na saída de `SHOW PROCESSLIST`, o número de segundos entre o timestamp do último evento replicado e o tempo real da máquina slave. Vopcê podee usar isto para determinar a data do último evento replicado. Note que se o

seu slave foi desconectado do master por uma hora e então reconectado, você poderá ver uma vlaor de 3600 na coluna `Time` para a thread slave de SQL em `SHOW PROCESSLIST`... Isto ocorreria porque o slave está executando consultas de uma hora atrás.

P: Como eu forço o master a bloquear as atualizações até que o slave as busque?

R: Use o seguinte procedimento:

1. No master, execute estes comandos:

   ```
   mysql> FLUSH TABLES WITH READ LOCK;
   mysql> SHOW MASTER STATUS;
   ```

   Grave o nome do log e o offset da saída da instução `SHOW`.

2. No slave, execute este comando, onde as coordenadas da replicação que são os argumentos da função `MASTER_POS_WAIT()` são os valores gravados nos passos anteriores:

   ```
   mysql> SELECT MASTER_POS_WAIT('log_name', log_offset);
   ```

   A instrução `SELECT` será bloqueada até que o slave alcance o arquivo de log e offset especificados. Neste ponto, o slave estará em sincronia com o master e a instrução irá retornar.

3. No master, execute a seguinte instrução para permitir que o master comece a processar atualizações novamente:

   ```
   mysql> UNLOCK TABLES;
   ```

P: Sobre quais assuntos eu devo estar ciente ao configurar uma replicação de duas vias?

R: Atualmente a replicação do MySQL não suporta nenhum protocolo de locking entre master e slave para garantir a atomicidade de uma atualização distribuída (entre servidores). Em outras palavras, é possível para um cliente A fazer uma atualização para um co-master 1, e neste tempo, antes de propagar para o co-master 2, o cliente B pode fazer uma atualização para o co-master 2 que fará a atualização do cliente A funcionar diferentemente da que ele fez no co-master 1. Assim, quando a atualização do cliente A fizer a atualização para o co-master, ele produzirá tabelas que são diferentes daquelas que você tem no co-master 1, mesmo depois de todas as atualizações do co-master2 também terem sido propagadas. Por isso você não deve co-encadear dois servidores em uma replicação de duas vias, a menos que você possa assegurar que suas atualizações possem seguramente ocorrer em qualquer ordem, ou que de alguma forma você cuide de alguma forma a atualização fora de ordem no código do cliente.

Você também deve perceber que replicação de duas vias não melhorar em muito o desempenho, já que temos atualizações envolvidas. Ambos os servidores precisam fazer a mesma quantidade de atualizações cada, como se tivesse um servidor. A única diferença é que haverá uma pouco menos de contenção de lock, porque as atualizações originando em outro servidor serão serializadas em uma thread slave. mesmo assim, este benefício pode ser por atrasos de rede.

P: Como eu posso usar replicação para melhorar a performance do meu sistema?

R: Você devev configurar um servidor como master e direcionar todas as escritas para ele. Então configure tantos slaves quantos você pode comprar e instalar, e distribua as leituras entre o master e os slaves. Você também pode iniciar os slaves com `--skip-bdb`, `--low-priority-updates` e `--delay-key-write=ALL` para conseguir aumento de velocidade para o slave. Neste caso o slave usará tabelas `MyISAM` não transacionais em vez de tabelas `BDB` para conseguir mais velocidade.

Q: O que eu devo fazer para preparar o código do cliente em minhas próprias aplicações para usar replicação para melhora de performance?

A: Se a parte do seu código que for responsável pela acesso ao banco de dados tiver sido abstaído/modularizado apropriadamente, converte-lo para executar com uma configuração de replicação deve ser bem fácil. Apenas altere a implementação de seu acesso a banco de dados para enviar todas as escritas para o master e todas as leituras para o master e o slave. Se o seu código não tiver este nível de abstração, configurar um sistema de replicação lhe dará a oportunidade e motivação de limpá-lo. Você deve iniciar criando uma biblioteca ou módulo wrapper com as seguites funções:

- `safe_writer_connect()`
- `safe_reader_connect()`
- `safe_reader_query()`
- `safe_writer_query()`

`safe_` no nome de cada função significa que a função cuidará do tratamento de todas as condições de erro.

Você pode, é claro, usar diferentes nomes para as funções. O importante é ter uma interface unificada para conexão para leitura, conexão para escrita, fazer uma leitura e fazer uma escrita.

Você deve então converter o código do seu cliente para usar a biblioteca wrapper. Este pode ser um processo doloroso e assustador a princípio, mas será gratificante a longo prazo. Todas as aplicações que usam a abordagem descrita poderão tirar vantagem de uma configuração master/slave, mesmo envolvendo vários slaves. O código será muito mais fácil de manter, e adicionar opções para soluções de problemas será trivial. Você só precisará modificar uma ou duas funções, por exemplo, para registrar quanto tempo uma consulta gastou ou qual consulta, entre todas elas, retornou um erro.

Se você já tiver escrito muito código, você pode querer automatizar a conversão de tarefas usando o utilitário `replace`, que vem com a distribuição padrão do MySQL, ou simplesmente escrever seu próprio script Perl. Provavelmente, o seu código segue algum padrão de reconhecimento. Se não, então talvez sejá melhor reescrevê-lo ou pelo menos colocá-lo dentro de um padrão.

Q: Quando e em quanto a replicação do MySQL pode aumentar a performance do meu sistema?

A: A replicação do MySQL é mais benéfica para um sistema com leituras frequentes e escritas infrequentes. Em teoria, usando uma configuração um mastre/vários slaves você pode escalar o sistema adicionando mais slaves até que você fique sem largura de banda na rede, ou a sua carga de atualizações cresça ao ponto que o master não possa tratá-la.

Para determinar quantos slaves você pode ter antes dos benefícios adicionados começarem a estabilzar e quanto você pode melhorar o desempenho do seu site, você precisa saber o padrão de suas consultas e determinar empiricamente (pelo benchmark) a ralação entre a taxa nas leituras (leituras por segundo, ou `max_reads`) e nas escritas (`max_writes`) em um master e um slave comum. O exemplo aqui lhe mostrará um calculo simplificado do que você pode obter com replicação para o nosso sistema hipotético.

Vamos dizer que o sistema de cargas consiste de 10% de escrita e 90% de leitura, e determinamos `max_reads` para ser 1200 - 2 * `max_writes`. Em outras palavras, nosso sistema

pode fazer 1200 leituras por segundo sem nenhuma escrita, a escrita média é duas vezes mais lenta que a leitura média e a realação é linear. Vamos supor que o master e cada slave possuem a mesma capacidade, e temos 1 master e N slaves. Então temos para cada servidor (master ou slave):

`leituras = 1200 - 2 * escritas` (a partir do benchmark)

`leituras = 9* escritas / (N + 1)` (as leituras são separadas, mas a escrita deve ir para todos os servidores)

`9*escritas/(N+1) + 2 * escritas = 1200`

`escritas = 1200/(2 + 9/(N+1)`

Esta análise leva as seguintes conclusões:

- Se N = 0 (que significa que não temos replicação) nosso sistema pode tratar 1200/11, cerca de 109, escritas por segundos (o que significa que teremos 9 vezes mais leituras devidos a natureza de nossa aplicação)
- Se N = 1, podemos aumentar para 184 escritas por segundos.
- Se N = 8, conseguimos 400.
- Se N = 17, 480 escritas.
- Eventualmente, a medida que N se aproxima de infinito (e seu orçamento de menos infinito), podemos chegar próximo a 600 escritas por segundo, aumentando o throughput do sistema em cerca de 5,5 vezes. No entanto, com apenas 8 servidores, já o aumentamos em quase 4 vezes.

Note que nossos calculos assumem uma largura de banda de rede infinita, e negligencia vários outros fatores que podiam se tornar significante em seu sistema. Em muitos casos, você pode não conseguir fazer um cálculo similar ao acima que irá predizer exatamente o que acontecerá em seus sistema se você adicionar N slaves de replicação. No entanto, responder as seguintes questões deve ajudá-lo a decidir quando e quanto a replicação aumentará a performance do seu sistema:

- Qual a razão da leitura/escrita no seu sistema?
- Quanto mais de carga de escrita um servidor pode tratar se você reduzir a leitura?
- Para quantos slaves você tem largura de banda disponíovel em sua rede?

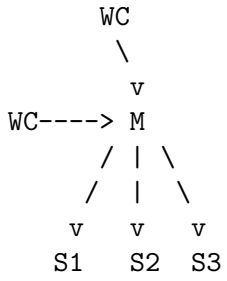
P: Como eu posso usar replicação para fornecer redundância/alta disponibilidade?

R: Com os recursos disponíveis atualmente, você teria que configurar um master e um slave (ou diversos slaves) e escrever um script que monitoraria o master para ver se ele está no ar e instruir as suas aplicações e os slaves do master a alterar no caso de falha. Sugestões:

- Para dizer para um slave para alterar o master, use o comando `CHANGE MASTER TO`.
- Um bom modo de manter sua aplicação informadas sobre a localização do master é tendo uma entrada DNS dinâmica para o master. Com `bind` você pode usar '`nsupdate`' para atualizar dinamicamente o seu DNS.
- Você deve executar seus escravos com a opção `--log-bin` e sem `--log-slave-updates`. Deste modo o slave estará pronto para se tornar um master assim que você executar `STOP SLAVE`; `RESET MASTER`, e `CHANGE MASTER TO` em outros slaves.

  Por exemplo, considere que voceê tenha a seguinte configuração ("M" representa o master, "S" o slave, "WC" o cliente que faz a leitura e escrita do banco de dados;

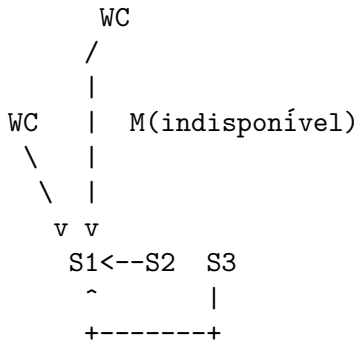
clientes que fazem apenas a leitura do banco de dados bão são representadas já que elas não precisam trocar):



S1 (como S2 e S3) é um slave executando com `--log-bin` e sem `--log-slave-updates`. Como as únicas escritas executada em S1 são aquelas replicadas de M, o log binário em S1 é **empty** (lembre-se, que S1 é executado sem `--log-slave-updates`). Então, por alguma razão, M se torna indisponível, e você quer que o S1 se torne o novo master (isto é, direciona todos os WC para S1 e faça S2 e S3 replicar em S1).

Certifique-se que todos os slaves processaram aqulquer consulta em seus relay log. Em cada slave, execute `STOP SLAVE IO_THREAD`, então verifique a saída de `SHOW PROCESSLIST` até você ver `Has read all relay log`. Quando isto ocorrer para todos os slaves, eles podem ser reconfigurados para a nova configuração. Envie `STOP SLAVE` para todos os slaves, `RESET MASTER` no slave sendo promovido para master, e `CHANGE MASTER` nos outros slaves.

Nenhum WC acessa M. Instru todos os WCs a direcionar as suas consultas para S1. De agora em diante, todas as consultas enviadas por WC para S1 são escritas no log binário de S1. O log binário de S1 contém exatamente todas as consultas de escrita enviada para S1 desde que M foi finalizado. Em S2 (e S3) faça `STOP SLAVE`, `CHANGE MASTER TO MASTER_HOST='S1'` (onde `'S1'` é substituido pelo nome de máquina real de S1). Para `CHANGE MASTER`, adicione todas as informações sobre como conectar a S1 de S2 ou S3 (usuário, senha, porta). Em `CHANGE MASTER`, não é necessário especificar o nome do log binário de S1 ou a sua posição: nós sabemos que ele é o primeiro log binário, na posição 4, e estes são os padrões de `CHANGE MASTER`. Finalmente faça `START SLAVE` em S2 e S3, e agora você terá isto:



Quando M estiver ativo novamente, você só precisa enviar a ele o mesmo `CHANGE MASTER` enviado a S2 e S3, assim que M se tornar um slave de S1 e pegar tudo que WC gravou enquando ele estava desativado. Agora para tornarmos M como master novamente (por exemplo, porque ela é a melhor máquina), siga os procedimentos como se S1 estivesse indisponível e M fosse o novo master; então durante o procedimento não esqueça d

executar `RESET MASTER` em M antes de tornar S1, S2, S3 como slaves de M ou eles podem buscar escritas antigas de WC, antes da indisponibilidade de M.

Atualmente estamos trabalhando na integração de um sistema de eleição de master autmotico dentro do MySQL, mas até que ele esteja pronto, você terá que criar suas próprias ferramentas de monitoramento.

4.11.10 Problemas com Replicação

Se você tiver seguido as instruções e suia configuração de replicação não está funcionando, primeiro verifique o seguinte:

- **Verifique as mensagens no log de erros**. Muitos usuários perderam tempo por não fazer isto cedo o suficiente.
- O master está logando ao log binário? Verifique com `SHOW MASTER STATUS`. Se estiver, `Position` será diferente de zero. Se não, verifique que deu a opção `log-bin` do master e definiu o `server-id`.
- O slave está executando? Faça `SHOW SLAVE STATUS` e verifique se os valores `Slave_IO_Running` e `Slave_SQL_Running` são ambos `Yes`. Se não, verifique a opção do slave.
- Se o slave estiver rodando, ele estabeleceu uma conexão com o master? Faça `SHOW PROCESSLIST`, encontre as threads de E/S e SQL (veja Seção 4.11.3 [Replication Implementation Details], Página 381 para ver como é exibido), e verifique a sua coluna `State`. Se ela disser `Connecting to master`, verifique os privilégios do usuário de replicação no master, nome de máquina do master, sua configuração de DNS, se o master está atualmente em execução e se ele está a alcance do slave.
- Se o slave estava em execução antes mas agora parou, a razão é que normalmente algumas consultas que obtem sucesso no master falham no slave. Into nunca deve acontecer se você tiver tirado a cópia apropriada do master e nunca modificou os dados no slave fora da thread slave. Se isto ocorrer, você encontrou um erro; leia abaixo como relatá-lo.
- Se uma consulta bem sucedida no master se recusou a executar no slave, e não parece prático fazer um nova sincronização completa do banco de dados (p.ex.: deletar o banco de dados slave e fazer uma nova cópia do master), tente o seguinte:
  - Primeiro veja se a tabela do slave estava diferente da do master. Entenda como isto aconteceu (pode ser um erro: leia o registro de alterações no manual online do MySQL como `http://www.mysql.com/documentation` para verificar se este é um erro conhecido e se ele já está corrigido). Então faça a tabela do slave idêntica a do master e execute `START SLAVE`.
  - Se o acima não funcionar ou não se aplica, tente entender se ele estaria seguro para fazer uma atualização manualmente (se necessário) e então ignorar a próxima consulta do master.
  - Se você decidiu que você pode saltar a próxima consulta, execute as seguintes instruções:

```
mysql> SET GLOBAL SQL_SLAVE_SKIP_COUNTER = n;
mysql> START SLAVE;
```

O valor de `n` deve ser 1 se a consulta não usa `AUTO_INCREMENT` ou `LAST_INSERT_ID()`. Senão, o valor de ser 2. A razão para usarem um valor de 2 para consultas que usam `AUTO_INCREMENT` ou `LAST_INSERT_ID()` é que elas gastam dois eventos no log binário do master.

- Tenha certeza de que você não está tendo problemas com um erro antigo atualizando para a versão mais recente.
- Se você tem certeza que o slave iniciou perfeitamente em sincronia com o master, e que as tabelas envolvidas não foram atualizadas fora da thread slave, relate o erro.

4.11.11 Relatando Problemas de Replicação

Quando você tiver determinado que não há erro de usuário envolvido, e a replicação ainda não funciona perfeitamente ou está instável, é hora de começar a fazer num relatório de erros. Nós precisamos do máximo de informações que você puder fornecer para conseguirmos rastrear o bug. Por favor gaste algum tempo e esforço preparando um bom relato de erro.

Se você tiver uma forma repetitível de demonstrar o problema, por favor inclua-o em nosso banco de dados de bugs `http://bugs.mysql.com`. Se você tem um problema de fantasma (um problema que não pode ser duplicado a sua vontade), use o seguinte procedimento:

1. Verifique se nenhum erro de usuário está envolvido. Por exemplo, se você atualiza o slave fora da thread slave, os dados podem ficar fora de sincronia e podem ocorrer violações de chave única nas atualizações. Neste caso a thread slave irá terminar e esperar que você limpe as tabelas manualmente para entrar em sincronia. Este não é um problema de replicação; é um problema de interferência externa que faz com que a replicação falhe.
2. Execute o slave com as opções `log-slave-updates` e `log-bin`. Elas farão com que o registre todas as atualizações que ele receber no seu próprio log binário.
3. Salve todas as evidências antes de restaurar o estado da replicação. Se não tivermos nenhuma informação ou apenas algum esboço, será um pouco mais difícil para rastrearmos o problema. As evidências que você deve coletar são:
   - Todos os logs binários no master
   - Todos os logs binários no slave
   - A saída de `SHOW MASTER STATUS` no master na hora que você descobriu o problema.
   - A saída de `SHOW SLAVE STATUS` no master na hora que você descobriu o problema.
   - Logs de erro no master e no slave
4. Utilize `mysqlbinlog` para examinar os logs binários. A informação a seguir pode ser útil para encontrar a consulta problemática, por exemplo:

```
mysqlbinlog -j pos_from_slave_status /caminho/para/log_do_slave | head
```

Uma vez que você coletou as evidências do problema fantasma, tente isolá-lo em um caso de testes separados inicialmente. Então relate o problema para `http://bugs.mysql.com/` com a maior quantidade possíveis de informações.

5 Otimização do MySQL

Otimização é uma tarefa complicada porque necessita um entendimento do sistema como um todo. Enquanto for possível fazer algumas otimizações com pequeno conhecimento de seu sistema ou aplicação, quanto mais otimizado você desejar que o seu sistema esteja, mais terá que saber sobre ele.

Este capítulo tentará explicar e fornecer alguns exemplos de diferentes formas de otimizar o MySQL. Lembre-se, no entanto, que sempre existirão (cada vez mais difíceis) formas adicionais de deixar seu sistema mais rápido.

5.1 Visão Geral da Otimização

A parte mais importante para obter um sistema rápido é com certeza o projeto básico. Você também precisa saber quais tipos de coisas seus sistema estará fazendo, e quais são gargalos existentes.

Os gargalos mais comuns são:

- Pesquisa em disco É necessário tempo para o disco encontrar uma quantidade de dados. Com discos modernos em 1999, o tempo médio para isto era normalmente menor que 10ms, portanto em teoria poderíamos fazer 100 buscas por segundo. Este tempo melhora moderadamente com discos novos e isso é muito difícil otimizar para uma única tabela. A maneira de otimizar isto é colocando os dados em mais de um disco.
- Leitura de disco/Escrita (I/O) Quando o disco estiver na posição correta precisaremos que os dados sejam lidos. Com discos mais modernos em 1999, um disco retorna algo em torno de 10-20Mb/s. Isto é mais fácil de otimizar que as buscas porque você pode ler vários discos em paralelo.
- Ciclos de CPU. Quando tivermos os dados na memória principal (ou se eles já estiverem lá) precisaremos processá-los para conseguir nosso resultado. O fator de limitação mais comum é ter ppequenas tabelas, comparadas com a memória. Mas, com pequenas tabelas, normalmente não teremos problemas com velocidade.
- Largura de banda da memória. Quando a CPU precisa de mais dados que podem caber no cache da CPU a largura da banda da memória principal se torna um gargalo. Isto é um gargalo muito incomum para a maioria dos sistema, mas é bom estarmos ciente dele.

5.1.1 Limitações do Projeto MySQL/Trocas

Quando usamos o mecanismos de armazenamento MyISAM, o MySQL utiliza travamento de tabela extremamente rápidos (múltiplas leituras / única escrita). O maior problema com este tipo de tabela ocorre quando você tem uma mistura do fluxo fixo de atualizações e seleções lentas na mesma tabela. Se isto for um problema com algumas tabelas, você pode usa outro tipo de tabela. Veja Cap“ptexi tulo 7 [Tipos de tabela], Página 630.

O MySQL pode trabalhar com tabelas transacionais e não transacionais. Para trabalhar sem problemas com tabelas não transacionais (nas quais não se pode fazer um rollback se alguma coisa der errada), o MySQL tem as seguintes regras:

- Todas as colunas possuem valor padrão.
- Se você inserir um valor 'errado' em uma coluna, como um `NULL` em uma coluna `NOT NULL` ou um valor numérico muito grande em uma coluna numérica, o MySQL definir a coluna com o 'melhor valor possível' em vez de dar um erro. Para valores numéricos isto é 0, o menor valor possível ou o maior valor possível. Para strings into é tanto uma string vazia quanto a maior string possível que possa estar na coluna.
- Todas as expressões calculadas retornam um valor que pode ser usado em vez de apresentar uma condição de erro. Por exemplo, 1/0 retorna `NULL`

Para mais informações sobre isto, veja Veja Seção 1.8.5 [Constraints], Página 52.

O mostrado acima quer dizer que não se deve usar o MySQL para verificar o conteúdo dos campos, mas deve se fazer isto no aplicativo.

5.1.2 Portabilidade

Como todos os servidores SQL implementam diferentes partes de SQL, é trabalhoso escrever aplicativos SQL portáveis. Para selects/inserts muito simples é muito fácil, mas quanto mais recursos você precisa, mais difícil se torna. Se você quiser uma aplicação quue é rápida com muitos bancos de dados ela se torna ainda mais difícil.

Para fazer um aplicativo portável complexo você precisa escolher um número de servidores SQL com o qual ele deve trabalhar.

Você pode utilizar o MySQL programa/web-page `crash-me` - `http://www.mysql.com/information/crash-` - para encontrar funções, tipos e limites que você pode utilizar com uma seleção de servidores de bancos de dados. O Crash-me agora testa quase tudo possível, mas continua compreensível com aproximadamente 450 itens testados.

Por exemplo, você não deve ter nomes de colunas maior do que 18 caracteres se desejar utilizar o Informix ou DB2.

Os programas de benchmarks e `crash-me` do MySQL são bastante independentes do bancos de dados. Dando uma olhada em como nós os tratamos, você pode sentir o que é necessário para escrever sua aplicação independente do banco de dados. Os benchmarks podem ser encontrados no diretório '`sql-bench`' na distribuição fonte do MySQL. Eles são escritos em Perl com a interface de banco de dados DBI (que resolve a parte do problema de acesso).

Veja `http://www.mysql.com/information/benchmarks.html` para os resultados deste benchmark.

Como pode ser visto nestes resultados, todos os bancos de dados tem alguns pontos fracos. Isto é, eles possuem diferentes compromissos de projeto que levam a comportamentos diferentes.

Se você procura por independencia de banco de dados, precisará ter uma boa idéia dos gargalos de cada servidor SQL. O MySQL é muito rápido para recuperação e atualização de dados, mas terá problemas em misturar leituras/escritas lentas na mesma tabela. O Oracle, por outro lado, possui um grande problema quando você tentar acessar registros que foram recentemente atualizados (até eles serem atualizados no disco). Bancos de dados transacionais geralmente não são muito bons gerando tabelas de resumo das tabelas log, nestes casos o travamento de registros é praticamente inútil.

Para fazer sua aplicação *realmente* independente de banco de dados, você precisará definir uma interface que possa ser expandida, por meio da qual você fará a manipulação dos dados. Como o C++ está disponível na maioria dos sistemas, faz sentido utilizar classes C++ para fazer a interface ao banco de dados.

Se você utilizar algum recurso específico para algum banco de dados (como o comando `REPLACE` no MySQL), você deve codificar um método para os outros serviodores SQL para implementar o mesmo recurso (mas mais lento). Com o MySQL você pode utilizar a sintaxe `/*! */` para adicionar palavras chave específicas do MySQL para uma query. O código dentro de `/**/` será tratado como um comentário (ignorado) pela maioria dos servidores SQL.

Se alta performance REAL é mais importante que exatidão, como em algumas aplicações WEB, uma possibilidade é criar uma camada de aplicação que armazena todos os resultados para lhe fornecer uma performance ainda mais alta. Deixando resultados antigos 'expirar' depois de um tempo, você pode manter o cache razoavelmente atual. Isto é muito bom no caso de uma carga extremamente pesada, pois neste caso você pode aumentar o cache dinamicamente e configurar o tempo de expiração maior até que as coisas voltem ao normal.

Neste caso a informação de criação de tabelas devem conter informações do tamanho inicial do cache e com qual frequência a tabela, normalmente, deve ser renovada.

5.1.3 Para que Utilizamos o MySQL?

Durante o desenvolvimento inicial do MySQL, os recursos do MySQL foram desenvolvidos para atender nosso maior cliente. Eles lidam com data warehousing para alguns dos maiores varejistas na Suécia.

De todas as lojas, obtemos resumos semanais de todas as transações de cartões de bonus e esperamos fornecer informações úteis para ajudar os donos das lojas a descobrir como suas campanhas publicitárias estão afetando seus clientes.

Os dados são bem grandes (cerca de 7 milhões de transações por mês), e armazenamos dados por cerca de 4-10 anos que precisamos apresentar para os usuários. Recebemos requisições semanais dos clientes que desejam ter acesso 'instantâneo' aos novos relatórios contendo estes dados.

Resolvemos este problema armazenando todas informações mensalmente em tabelas com transações compactadas. Temos um conjunto de macros (script) que geram tabelas resumidas agrupadas por diferentes critérios (grupo de produto, id do cliente, loja...) das tabelas com transações. Os relatórios são páginas Web que são geradas dinamicamente por um pequeno shell script que analisa uma página Web, executa as instruções SQL na mesma e insere os resultados. Nós usariamos PHP ou mod_perl mas eles não estavam disponíveis na época.

Para dados graficos escrevemos um ferramenta simples em `C` que pode produzir GIFs baseados no resultado de uma consulta SQL (com alguns processamentos do resultado). Isto também é executado dinamicamente a partir do script Perl que analisa os arquivos HTML.

Na maioria dos casos um novo relatório pode simplesmente ser feito copiando um script existente e modificando a consulta SQL no mesmo. Em alguns casos, precisamos adicionar mais campos a uma tabela de resumo existente ou gerar uma nova, mas isto também é bem

simples, pois mantemos todas as tabelas com as transaçõs no disco. (Atualmente possuimos pelo menos 50G de tabelas com transações e 200G de outos dados do cliente.)

Nós também deixamos nossos clientes acessarem as tabelas sumárias diretamente com ODBC para que os usuários avançados possam também fazer experimentar com os dados.

Nós não tivemos nenhum problema lidando com isso em um servidor Sun Ultra SPARC-station (2x200 Mhz) bem modesto. Atualmente atualizamos um de nossos servidores para um UltraSPARC com 2 CPUs de 400 Mhz, e planejamos lidar com transações no nível de produto, o que pode significar um aumento de pelo menos dez vezes nosso volume de dados. Acreditamos que podemos lidar com isto apenas adicionando mais disco aos nossos sistemas.

Também estamos experimentando com Intel-Linux para obter mais poder de CPU por um melhor preço. Agora que possuimos o formato binários do bancos de dados portáveis (a partir da versão 3.23), começaremos a utilizá-lo para partes da aplicação.

Nossa sensação inicial é que o Linux irá atuar muito melhor em cargas baixas e médias e o Solaris irá atuar melhor quando você começar a ter uma carga alta pelo uso extremo de IO de disco, mas ainda não temos nada conclusivo sobre isto. Depois de algumas discussões com um desenvolvedor do kernel do Linux, concluímos que isto pode ser um efeito colateral do Linux; alocar muitos recursos para uma tarefa batch que a performance interativa se torna muito baixa. Isto deixa a máquina muito lenta e sem resposta enquanto grandes batches estiverem em execução. Esperamos que isto tenha um tratamento melhor em futuras versões do kernel Linux.

5.1.4 O Pacote de Benchmark do MySQL

Esta seção deve conter uma descrição técnica do pacote de benchmarks do MySQL (e `crash-me`), mas a descrição ainda não está pronta. Atualmente, você pode ter uma boa idéia do benchmark verificando os códigos e resultados no diretório '`sql-bench`' em qualquer distribuição fonte do MySQL.

Este conjunto de benchmark pretende ser um benchmark que irá dizer a qualquer usuário que operações uma determinada implementação SQL irá realizar bem ou mal.

Note que este benchmark utiliza uma única thead, portanto ele mede o tempo mínimo para as operações realizadas. Planejamos adicionar vários testes multi-threaded no conjunto de benchmark no futuro.

A seguinte tabela mostra alguns resultados comparativos de benchmark para diversos servidores de bancos de dados quando acessados por meio do ODBC em uma máquina Windows NT 4.0.

Lendo 2000000 linhas por índice	**Segundos**	**Segundos**
mysql	367	249
mysql_odbc	464	
db2_odbc	1206	
informix_odbc	121126	
ms-sql_odbc	1634	
oracle_odbc	20800	
solid_odbc	877	
sybase_odbc	17614	

Inserindo 350768 linhas	**Segundos**	**Segundos**
mysql	381	206
mysql_odbc	619	
db2_odbc	3460	
informix_odbc	2692	
ms-sql_odbc	4012	
oracle_odbc	11291	
solid_odbc	1801	
sybase_odbc	4802	

Para os testes anteriores, o MySQL foi executado com um cache de índices de 8M.

Temos concentrado alguns resultados de benchmarks em `http://www.mysql.com/information/benchmark`

Perceba que a Oracle não está incluída porque eles solicitaram a remoção. Todos benchmarks Oracle devem ser aprovados pela Oracle! Acreditamos que os benchmarks da Oracle são **MUITO** tendecioso pois os benchmarks acima devem ser executados supostamente para uma instalação padrão para um único cliente.

Para executar a suite de benchmarks, as seguintes exigências devem ser satisfeitas:

- O pacote de benchamark é fornecido com a distribuição fonte do MySQL, assim você deve ter uma distribuição fonte. Você também pode fazer um download de uma distribuição em `http://www.mysql.com/downloads/`, ou usar a árvore fonte de desenvolvimento atual. (veja Seção 2.3.4 [Installing source tree], Página 100).
- Os scripts do benchmark são escritos em Perl e usam o módulo Perl DBI para acessar o servidor de banco de dados, assim o DBI deve estar instalado. Você também precisará do driver DBD espercífico do servidor para cada um dos servidores que você quer testar. Por exemplo, para testar o MySQL, PostgreSQL, e DB2, os módulos DBD::mysql, DBD::Pg e DBD::DB2 devem estar instalados.

O pacote de benchmark está localizado no diretório '`sql-bench`' da distribição fonte do MySQL. Para executar o teste de benchmark, altera a localização dentro daquele diretório e execute o script `run-all-tests`:

```
shell> cd sql-bench
shell> perl run-all-tests --server=server_name
```

`server_name` é um dos servidores suportados. Você pode obter uma lista de todos parâmetros e servidores suportados executando `run-all-tests --help`.

`crash-me` tenta determinar quais recursos um banco de dados suporta e quais suas capacidades e limitações atuais para a execução de consultas. Por exemplo, ele determina:

- Quais tipos de colunas são suportados
- Quantos índices são suportados
- Quais funções são suportadas
- Qual o tamanho máximo de uma query
- Qual o tamanho máximo de um registro do tipo `VARCHAR`

Podemos encontrar o resultado do `crash-me` para diversos bancos de dados em `http://www.mysql.com/information/crash-me.php`.

5.1.5 Utilizando seus Próprios Benchmarks

Definitivamente você deve fazer benchmarks de sua aplicação e banco de dados para saber quais são os gargalos. Corrigindo (ou substituindo o gargalho com um "módulo burro") você pode facilmente identificar o próximo gargalo (e continuar). Mesmo se a performance geral para sua aplicação atualmente é aceitável, você deve pelo menos criar um plano para cada gargalo e decidir como resolvê-lo se algum dia você precisar de performance extra.

Para um exemplo de programas de benchmarks portáveis, consulte o conjunto de benchmarks do MySQL. Veja Seção 5.1.4 [Benchmarks do MySQL], Página 421. Você pode pegar qualquer programa deste conjunto e modificá-lo para suas necessidades. Fazendo isto você pode tentar soluções diferentes para seu problema e testar qual é a mais rápida para você.

Outro pacote de benchmark grátis é o `Open Source Database Benchmark` disponível em `http://osdb.sourceforge.net/`.

É muito comum que um problemas ocorram apenas quando o sistema estiver muito carregado. Nós tivemos alguns clientes que nos contactaram quando eles testaram um sistema em produção e encontraram problemas de carga. Na maioria dos casos, problemas de desempenho ocorrem devido a assuntos relacionados ao projeto básico do banco de dados (busca em tabelas `não são bons` com alta carga) ou problemas com o sistema operacional e de bibliotecaa. A maioria das vezes, estes problemas seriam **MUITO** mais fáceis de resolver se os sistemas já não estivessem em uso.

Para evitar problemas deste tipo, você deve colocar algum esforço em testar a performance de toda sua aplicação sobre a pior carga possível! Você pode utilizar o Super Smack para isto. Ele está disponível em: `http://www.mysql.com/Downloads/super-smack/super-smack-1.0.tar.gz`.
Como o nome sugere, ele pode derrubar seu sistema se você solicitar, portanto, utilize-o somente em sistemas de desenvolvimento.

5.2 Otimizando `SELECT`s e Outras Consultas

Primeiramente, uma coisa que afeta todas as consultas: Quanto mais complexo seu sistema de permissões, maior a sobrecarga.

Se você não tiver nenhuma instrução `GRANT` realizada, MySQL otmizará a verificação de permissões de alguma forma. Dessa forma, se você possui um volume muito alto, o tempo pode piorar tentando permitir o acesso. Por outro lado, maior verificação de permissões resulta em uma sobrecarga maior.

Se o seu problema é com alguma função explícita do MySQL, você pode sempre consultar o tempo da mesma com o cliente MySQL:

```
mysql> SELECT BENCHMARK(1000000,1+1);
+------------------------+
| BENCHMARK(1000000,1+1) |
+------------------------+
|                      0 |
+------------------------+
1 row in set (0.32 sec)
```

O exemplo acima demonstra que o MySQL pode excutar 1.000.000 expressões `+` em 0.32 segundos em um `PentiumII 400MHz`.

Todas funções MySQL devem ser bem otimizadas, mas existem algumas excessões e o `benchmark(loop_count,expression)` é uma ótima ferramenta para saber se existe um problema com sua query.

5.2.1 Sintaxe de `EXPLAIN` (Obter informações sobre uma `SELECT`)

```
        EXPLAIN nome_tabela
    ou  EXPLAIN SELECT opções_select
```

`EXPLAIN nome_tabela` é um sinônimo para `DESCRIBE nome_tabela` ou `SHOW COLUMNS FROM nome_tabela`.

Quando uma instrução `SELECT` for precedida da palavra chave `EXPLAIN`, o MySQL explicará como ele deve processar a `SELECT`, fornecendo informação sobre como as tabelas estão sendo unidas e em qual ordem.

Com a ajuda de `EXPLAIN`, você pode ver quando devem ser adicionados índices à tabelas para obter uma `SELECT` mais rápida que utiliza índices para encontrar os registros.

Voce deve executar frequentemente `ANALYZE TABLE` para atualizar estatísticas de tabela tais como a cardinalidade das chaves que podem afetar a escolha que o otimizador faz. Veja Seção 4.6.2 [ANALYZE TABLE], Página 298.

Você também pode ver se o otimizador une as tabelas em uma melhor ordem. Para forçar o otimizador a utilizar uma ordem específica de join para uma instrução `SELECT`, adicione uma cláusula `STRAIGHT_JOIN`.

Para ligações mais complexas, `EXPLAIN` retorna uma linha de informação para cada tabela utilizada na instrução `SELECT`. As tabelas são listadas na ordem que seriam lidas. O MySQL soluciona todas as joins utilizando um método multi-join de varedura simples. Isto significa que o MySQL lê uma linha da primeira tabela, depois encontra uma linha que combina na segunda tabela, depois na terceira tabela e continua. Quando todas tabelas são processadas, ele exibe as colunas selecionadas e recua através da lista de tabelas até uma tabela na qual existem registros coincidentes for encontrada. O próximo registro é lido desta tabela e o processo continua com a próxima tabela.

No MySQL versão 4.1 a saída do `EXPLAIN` foi alterada para funcionar melhor com construções como `UNION`s, subqueries e tabelas derivadas. A mais notável é a adição de duas novas colunas: `id` e `select_type`.

A saída de `EXPLAIN` inclui as seguintes colunas:

`id` Identificador `SELECT`, o número sequêncial desta `SELECT` dentro da consulta.

`select_type`
Tipo de cláusula `SELECT`, que pode ser uma das seguintes:

- `SIMPLE` `SELECT` simples (sem `UNION`s ou subqueries).
- `PRIMARY` `SELECT` mais externa.
- `UNION` Segunda `SELECT` e as `SELECT`s posteriores do `UNION`

`DEPENDENT UNION`
: Seunda `SELECT` e `SELECT`s posteriores do `UNION`, dependente da subquery exterior.

`SUBQUERY` Primeiro `SELECT` na subquery.

`DEPENDENT SUBQUERY`
: Primeiro `SELECT`, dependente da subquery exterior.

`DERIVED` `SELECT` de tabela derivada (subquery na cláusula `FROM`).

`table` A tabela para a qual a linha de saída se refere.

`type` O tipo de join. Os diferentes tipos de joins são listados aqui, ordenados do melhor para o pior tipo:

`system` A tabela só tem uma linha (= tabela de sistema). Este é um caso especial do tipo de join `const`.

`const` A tabela têm no máximo um registro coincidente, o qual será lido na inicialização da consulta. Como só há um registro, os valores da coluna neste registro podem ser considerados constantes pelo resto do otimizador. Tabelas `const` são muito rápidas e são lidas apenas uma vez!

`const` é usado quando você compara todas as partes de uma chave `PRIMARY/UNIQUE` com restrições:

```
SELECT * FROM const_table WHERE primary_key=1;

SELECT * FROM const_table
WHERE primary_key_part1=1 AND primary_key_part2=2;
```

`eq_ref` Uma linha será lida desta tabela para cada combinação de linhas da tabela anterior. Este é o melhor tipo de join depois dos tipos `const`. É usado quando todas as partes do índice são usados pela join e o índice é é único (`UNIQUE`) ou uma chave primária (`PRIMARY KEY`).

`eq_ref` pode ser usado para coluna indexadas que é comparada com o\ operador `=`. O item comparado pode ser uma constante ou uma expressão que usa colunas de tabelas que são lidas antes desta tabela.

Nos seguintes examplos, `ref_table` poderá usar `eq_ref`

```
SELECT * FROM ref_table,other_table
WHERE ref_table.key_column=other_table.column;

SELECT * FROM ref_table,other_table
WHERE ref_table.key_column_part1=other_table.column
AND ref_table.key_column_part2=1;
```

`ref` Todas as colunas com valores de índices correspondentes serão lidos desta tabela para cada combinação de registros da tabela anterior. `ref` é usado se o join usa apenas o prefixo mais a esquerda da

chave, ou se a chave não é única (`UNIQUE`) ou uma chave primária (`PRIMARY KEY`) (em outras palavras, se a join não puder selecionar um único registro baseado no valor da chave). Se a chave que é usada coincide apenas em alguns registros, este tipo de join é bom.

`ref` pode ser usado para colunas indexadas que são comparadas com o operador `=`.

Nos seguintes exemplos, `ref_table` poderá usar `ref`

```
SELECT * FROM ref_table WHERE key_column=expr;

SELECT * FROM ref_table,other_table
WHERE ref_table.key_column=other_table.column;

SELECT * FROM ref_table,other_table
WHERE ref_table.key_column_part1=other_table.column
AND ref_table.key_column_part2=1;
```

`ref_or_null`

Como `ref`, mas com o adicional que faremos uma busca extra para linhas com `NULL`. Veja Seção 5.2.5 [IS NULL optimization], Página 434.

```
SELECT * FROM ref_table WHERE key_column=expr OR key_column IS
```

Esta otimização do tipo join é nova para o MySQL 4.1.1 e é mais usada na resolução de sub queries.

`range` Apenas registros que estão numa dada faixa serão retornados, usando um índice para selecionar os registros. A coluna `key` indica qual índice é usado. `key_len` contém a maior parte da chave que foi usada. A coluna `ref` será `NULL` para este tipo.

`range` pode ser usado para quando uma coluna de chave é comparada a uma constante com `=`, `<>`, `>`, `>=`, `<`, `<=`, `IS NULL`, `<=>`, `BETWEEN` e `IN`.

```
SELECT * FROM range_table WHERE key_column = 10;

SELECT * FROM range_table WHERE key_column BETWEEN 10 and 20;

SELECT * FROM range_table WHERE key_column IN (10,20,30);

SELECT * FROM range_table WHERE key_part1= 10 and key_part2 IN
```

`index` Isto é o mesmo que `ALL`, exceto que apenas a árvore de índice é varrida. Isto é normalmente mais rápido que `ALL`, já que o arquivo de índice normalmente é menor que o arquivo de dados.

Ele pode ser usado quando a consulta só usa colunas que são parte de um índice.

`ALL` Será feita uma varredura completa da tabela para cada combinação de registros da tabela anterior. Isto normalmente não é bom se a

tabela é a primeiro tabela não marcada como `const`, e normalmente **muito** ruim em todos os casos ordenados. Você normalmente pode ebitar `ALL` adicionando mais índices, assim o registro pode ser retornado baseado em valores constantes ou valores de colunas de tabelas anteriores.

`possible_keys`

A coluna `possible_keys` indica quais índices o MySQL pode utilizar para encontrar os registros nesta tabela. Note que esta coluna é totalmente independente da ordem das tabelas. Isto significa que algumas das chaves em `possible_keys` podem não ser usadas na prática com a ordem de tabela gerada.

Se esta coluna for `NULL`, não existem índices relevantes. Neste caso, você poderá melhora a performance de sua query examinando a cláusula `WHERE` para ver se ela refere a alguma coluna ou colunas que podem ser indexadas. Se for verdade, crie um índice apropriado e confira a consulta com `EXPLAIN` novamente. Veja Seção 6.5.4 [`ALTER TABLE`], Página 608.

Para ver os índices existentes em uma tabela, utilize `SHOW INDEX FROM nome_tabela`.

`key` A coluna `key` indica a chave (índice) que o MySQL decidiu usar. A chave será `NULL` se nenhum índice for escolhido. Para forçar o MySQL a usar um índice listado na coluna `possible_keys`, use `USE INDEX/IGNORE INDEX` em sua consulta. Veja Seção 6.4.1 [`SELECT`], Página 562.

Executando `myisamchk --analyze` (veja Seção 4.5.6.1 [sintaxe do `myisamchk`], Página 281) ou `ANALYSE TABLE` (veja Seção 4.6.2 [`ANALYZE TABLE`], Página 298) na tabela também ajudará o otimizador a escolher índices melhores.

`key_len` A coluna `key_len` indica o tamanho da chave que o MySQL decidiu utilizar. O tamanho será `NULL` se `key` for `NULL`. Note que isto nos diz quantas partes de uma chave multi-partes o MySQL realmente está utilizando.

`ref` A coluna `ref` exibe quais colunas ou contantes são usadas com a `key` para selecionar registros da tabela.

`rows` A coluna `rows` informa o número de linhas que o MySQL deve examinar para executar a consulta.

`Extra` Esta coluna contem informações adicionais de como o MySQL irá resolver a consulta. A seguir uma explicação das diferentes strings de texto que podem ser encontradas nesta coluna:

`Distinct` O MySQL não continuará a procurar por mais registros para a combinação de registro atual depois de ter encontrado o primeiro registro coincidente.

`Not exists`

O MySQL estava apto a fazer uma otimização `LEFT JOIN` na consulta e não examinará mais registros nesta tabela para a combinação do registro anterior depois que encontrar um registro que satisfaça o critério do `LEFT JOIN`.

Exemplo:

```
SELECT * FROM t1 LEFT JOIN t2 ON t1.id=t2.id
WHERE t2.id IS NULL;
```

Assume que `t2.id` é definido com `NOT NULL`. Neste caso o MySQL irá percorrer `t1` e procurar pelos registros em `t2` através de `t1.id`. Se o MySQL encontrar um registro combinando em `t2`, ele sabe que `t2.id` nunca poderá ser `NULL` e não ir percorrer até o resto dos registros em `t2` que possuirem o mesmo `id`. Em outras palavras, para cada registro em `t1` o MySQL só precisa fazer uma única pesquisa em `t2`, independente de quantos registros coincidentes existirem em `t2`.

`range checked for each record (index map: #)`
: O MySQL não encontrou um bom índice para usar. No lugar, ele irá fazer uma verificação sobre qual índice usar (se existir) para cada combinação das tabelas precedentes, e usará este índice para recuperar os registros da tabela. Isto não é muito rápido mas é mais rápido que fazer um join sem um índice.

`Using filesort`
: O MySQL precisará fazer uma passada extra para descobrir como recuperar os registros na ordem de classificação. A classificação é feita indo através de todos os registros de acordo com `join type` e armazenar a chave de ordenação mais o ponteiro para o registro para todos os registros que combinarem com o `WHERE`. Então as chaves são classificadas. Finalmente os registros são recuperados na ordem de classificação.

`Using index`
: A informação da coluna é recuperada da tabela utilizando somente informações na árvore de índices sem ter que fazer uma pesquisa adicional para ler o registro atual. Isto pode ser feito quando todas as colunas usadas para a tabela fizerem parte do mesmo índice.

`Using temporary`
: Para resolver a consulta, o MySQL precisará criar uma tabela temporária para armazenar o resultado. Isto acontece normalmente se você fizer um `ORDER BY` em um conjunto de colunas diferentes das quais você fez um `GROUP BY`.

`Using where`
: Uma cláusula `WHERE` será utilizada para restringir quais registros serão combinados com a próxima tabela ou enviar para o cliente. se você não possui esta informação e a tabela é do tipo `ALL` ou `index`, pode existir alguma coisa errada na sua query (Se você não pretender examinar todos os registros da tabela).

Se você desejar deixar suas consultas o mais rápido possível, você deve dar uma olhada em `Using filesort` e `Using temporary`.

Você pode ter uma boa indicação de quão boa é sua join multiplicando todos os valores na coluna `rows` na saída de `EXPLAIN`. Isto deve dizer a grosso modo quantos registros o

MySQL deve examinar para executar a consulta. Este número é também usado quando você restringe consultas com a variável `max_join_size`. Veja Seção 5.5.2 [Parâmetros de servidor], Página 454.

O exemplo a seguir mostra como um `JOIN` pode ser otimizado progressivamente utilizando a informação fornecida por `EXPLAIN`.

Suponha que você tem a instrução `SELECT` exibida abaixo, que você está examinando utilizando `EXPLAIN`:

```
EXPLAIN SELECT tt.TicketNumber, tt.TimeIn,
            tt.ProjectReference, tt.EstimatedShipDate,
            tt.ActualShipDate, tt.ClientID,
            tt.ServiceCodes, tt.RepetitiveID,
            tt.CurrentProcess, tt.CurrentDPPerson,
            tt.RecordVolume, tt.DPPrinted, et.COUNTRY,
            et_1.COUNTRY, do.CUSTNAME
        FROM tt, et, et AS et_1, do
        WHERE tt.SubmitTime IS NULL
            AND tt.ActualPC = et.EMPLOYID
            AND tt.AssignedPC = et_1.EMPLOYID
            AND tt.ClientID = do.CUSTNMBR;
```

Para este exemplo, assuma que:

- As colunas comparadas foram declaradas como a seguir:

Tabela	Coluna	Tipo da coluna
tt	ActualPC	CHAR(10)
tt	AssignedPC	CHAR(10)
tt	ClientID	CHAR(10)
et	EMPLOYID	CHAR(15)
do	CUSTNMBR	CHAR(15)

- As tabelas possuem os índices mostrados abaixo:

Tabela	Índice	
tt	ActualPC	
tt	AssignedPC	
tt	ClientID	
et	EMPLOYID	(chave primária)
do	CUSTNMBR	(chave primária)

- The `tt.ActualPC` values aren't evenly distributed.

Initially, before any optimizations have been performed, the `EXPLAIN` statement produces the following information:

```
table type possible_keys                key  key_len ref  rows  Extra
et    ALL  PRIMARY                      NULL NULL    NULL 74
do    ALL  PRIMARY                      NULL NULL    NULL 2135
et_1  ALL  PRIMARY                      NULL NULL    NULL 74
tt    ALL  AssignedPC,ClientID,ActualPC NULL NULL    NULL 3872
```

```
        range checked for each record (key map: 35)
```

Como o `tipo` é `ALL` em todas tabelas, esta saída indica que o MySQL está gerando um produto Cartesiano de todas as tabelas! Isto levará muito tempo para ser executado, pois o produto do número de registros em cada tabela deve ser examinado ! Neste caso, existem `74 * 2135 * 74 * 3872` registros. Se as tabelas forem maiores, imagine quanto tempo este tipo de consulta pode demorar.

Um dos problemas aqui é que o MySQL não pode (ainda) utilizar índices em colunas de maneira eficiente se elas foram declaras ide forma diferente. Neste contexto, `VARCHAR` e `CHAR` são o mesmo a menos que tenham sido declarados com tamanhos diferentes. Como `tt.ActualPC` é declarado como `CHAR(10)` e `et.EMPLOYID` é declarado como `CHAR(15)`, existe aqui uma diferença de tamanho.

Para corrigir esta diferença entre tamanhos de registros, utilize `ALTER TABLE` para alterar o tamanho de `ActualPC` de 10 para 15 caracteres:

```
mysql> ALTER TABLE tt MODIFY ActualPC VARCHAR(15);
```

Agora ambos campos `tt.ActualPC` e `et.EMPLOYID` são `VARCHAR(15)`. Executando a instrução `EXPLAIN` novamente produzirá este resultado:

```
table type   possible_keys   key     key_len ref         rows    Extra
tt    ALL    AssignedPC,ClientID,ActualPC NULL NULL NULL 3872    Using where
do    ALL    PRIMARY         NULL    NULL    NULL        2135
      range checked for each record (key map: 1)
et_1  ALL    PRIMARY         NULL    NULL    NULL        74
      range checked for each record (key map: 1)
et    eq_ref PRIMARY         PRIMARY 15      tt.ActualPC 1
```

Isto não está perfeito, mas está bem melhor (o produto dos valores de `rows` agora menor por um fator de 74). Esta versão é executada em vários segundos.

Uma segunda alteração pode ser feita para eliminar as diferenças de tamanho das colunas para as comparações `tt.AssignedPC = et_1.EMPLOYID` e `tt.ClientID = do.CUSTNMBR` :

```
mysql> ALTER TABLE tt MODIFY AssignedPC VARCHAR(15),
    ->                MODIFY ClientID   VARCHAR(15);
```

Agora `EXPLAIN` produz a saída mostrada abaixo:

```
table type   possible_keys   key      key_len ref           rows Extra
et    ALL    PRIMARY         NULL     NULL    NULL          74
tt    ref    AssignedPC,     ActualPC 15      et.EMPLOYID   52   Using where
             ClientID,
             ActualPC
et_1  eq_ref PRIMARY         PRIMARY  15      tt.AssignedPC 1
do    eq_ref PRIMARY         PRIMARY  15      tt.ClientID   1
```

Este resultado é quase o melhor que se pode obter.

O problema restante é que, por padrão, o MySQL assume que valores na coluna `tt.ActualPC` estão distribuídos igualmente, e este não é o caso para a tabela `tt`. Felizmente, é fácil informar ao MySQL sobre isto:

```
shell> myisamchk --analyze PATH_TO_MYSQL_DATABASE/tt
shell> mysqladmin refresh
```

Agora a join está perfeita, e `EXPLAIN` produz esta saída:

```
table type   possible_keys key     key_len ref           rows Extra
tt    ALL    AssignedPC    NULL    NULL    NULL          3872 Using where
             ClientID,
             ActualPC
et    eq_ref PRIMARY       PRIMARY 15      tt.ActualPC   1
et_1  eq_ref PRIMARY       PRIMARY 15      tt.AssignedPC 1
do    eq_ref PRIMARY       PRIMARY 15      tt.ClientID   1
```

Perceba que a coluna `rows` na saída de `EXPLAIN` é uma boa ajuda para otimizador de joins do MySQL. Para otimizar uma consulta, você deve conferir se os números estão perto da realidade. Se não, você pode obter melhor desempenho utilizando `STRAIGHT_JOIN` em sua instrução `SELECT` e tentar listar as tabelas em uma ordem diferente na cláusula `FROM`.

5.2.2 Estimando o Desempenho de uma Consulta

Na maioria dos casos você pode estimar a performance contando buscas em disco. Para tabelas pequenas, normalmente você pode encontrar o registro com 1 pesquisa em disco (uma vez que o índice provavelmente está no cache). Par tabelas maiores, você pode estimar (usando índices de arvores B++) que você precisará de: `log(row_count) / log(index_block_length / 3 * 2 / (index_length + data_pointer_length)) + 1` buscas em disco para encontrar um registro.

No MySQL um bloco de índice tem geralmente 1024 bytes e o ponteiro de dados 4 bytes. Uma tabela de 500.000 registros com um índice com tamanho de 3 (inteiro médio) lhe dá: `log(500,000)/log(1024/3*2/(3+4)) + 1` = 4 pesquisas.

Como o índice acima necessita cerca de 500,000 * 7 * 3/2 = 5.2M, (assumindo que os buffers de índices são carregados até 2/3, que é o normal) você provavelmente terá grande parte dos índices em memória e provavelmente precisará somente de 1 ou 2 chamadas para ler dados do SO para encontrar o registro.

Entretanto, para escritas, você precisará utilizar 4 requisições para encontrar onde posicionar o novo índice e normalmente 2 buscas para atualizar o índice e escrever o registro.

Perceba que o que foi dito acima não significa que sua aplicação perderá performance por N log N! Como tudo é armazenado no cache de seu SO ou do servidor SQL as coisas começarão a ficar um pouco mais lentas quando as tabelas começarem a crescer. Quando os dados se tornam muito grandes para o cache, as coisas começarão a ficar bem mais lentas até que suas aplicações estejam limitadas a buscas em disco (o que aumenta em N log N). Para evitar isto, aumente o cache de índice quando os dados crescerem. Veja Seção 5.5.2 [Parâmetros do servidor], Página 454.

5.2.3 Velocidade das Consultas que Utilizam `SELECT`

Em geral, quando você desejar tornar uma consulta lenta `SELECT ... WHERE` mais rápida, a primeira coisa que deve ser conferida é se você pode ou não adicionar um índice. Veja Seção 5.4.3 [índices MySQL], Página 447. Todas as referências entre diferentes tabelas devem ser feitas normalmente com índices. Você pode utilizar o comando `EXPLAIN` para determinas quais índices são usados para uma `SELECT`. Veja Seção 5.2.1 [`EXPLAIN`], Página 424.

Algumas dicas gerais:

- Para ajudar o MySQL a otimizar melhor as consultas, execute `myisamchk --analyze` em uma tabela depois dela ter sido carregada com dados relevantes. Isto atualiza um valor para cada parte do índice que indica o número médio de registros que tem o mesmo valor. (Para índices únicos, isto é sempre 1, é claro). O MySQL usará isto para decidir qual índice escolher quando você conectar duas tabelas utilizando uma 'expressão não constante'. Os resultados de `analyze` podem ser conferidos utilizando `SHOW INDEX FROM nome_tabela` e examindo a coluna `Cardinality`.
- Para ordenar um índice e dados de acordo com um índice, utilize `myisamchk --sort-index --sort-records=1` (se você deseja ordenar pelo índice 1). Se você possui um índice unico no qual deseja ler todos registros na ordem do índice, esta é uma boa forma para torná-lo mais rápido. Perceba entretanto, que esta ordenação não foi escrita de maneira otimizada e levará muito tempo em tabelas grandes!

5.2.4 Como o MySQL Otimiza Cláusulas `WHERE`

As otimizações `WHERE` são colocadas aqui na parte da `SELECT` porque normalmente elas são usadas com `SELECT`, mas as mesmas otimizações aplicam-se para `WHERE` em instruções `DELETE` e `UPDATE`.

Note também que esta seção está incompleta. O MySQL faz várias otimizações e ainda não tivemos tempo para documentarmos todas elas.

Algumas das otimizações feitas pelo MySQL são são listadas abaixo:

- Remoção de parênteses desnecessários:

```
   ((a AND b) AND c OR (((a AND b) AND (c AND d))))
-> (a AND b AND c) OR (a AND b AND c AND d)
```

- Enlaços de constantes:

```
   (a<b AND b=c) AND a=5
-> b>5 AND b=c AND a=5
```

- Remoção de condições contantes (necessário por causa dos enlaços de contantes):

```
   (B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)
-> B=5 OR B=6
```

  Expressões constantes utilizadas por índices são avaliadas somente uma vez.
- `COUNT(*)` em uma única tabela sem um `WHERE` é recuperado diretamente da informação da tabela dos tipos `MyISAM` e `HEAP`. Isto também é feito para qualquer expressão `NOT NULL` quando usada somente com uma tabela.
- Pré detecção de expressões contantes inválidas. O MySQL detecta rapidamente que algumas instruções `SELECT` são impossíveis e não retornará registros.
- `HAVING` é fundido com `WHERE` se não for utilizado `GROUP BY` ou funções de agrupamento (`COUNT()`, `MIN()`...).
- Para cada sub-join, um `WHERE` mais simples é construído para obter uma avaliação mais rápida de `WHERE` para cada sub-join e também para saltar registros da maneira mais rápida possível.
- Todas tabelas constantes são lidas primeiro, antes de qualquer tabelas na consulta. Uma tabela constante é:

  - Uma tabela vazia ou uma tabela com 1 registro.
  - Uma tabela que é usada com uma cláusula `WHERE` em um índice `UNIQUE`, ou uma `PRIMARY KEY`, onde todas as partes do índice são usadas com expressões constantes e as partes do índice são definidas como `NOT NULL`.

  Todas as tabelas seguintes são usadas como tabelas constantes:

```
mysql> SELECT * FROM t WHERE primary_key=1;
mysql> SELECT * FROM t1,t2
    ->          WHERE t1.primary_key=1 AND t2.primary_key=t1.id;
```

- A melhor combinação de join para unir as tabelas é encontrada tentando todas as possibilidades. Se todas colunas em `ORDER BY` e em `GROUP BY` vierem da mesma tabela, então esta tabela será preferencialmente a primeira na união.
- Se existerem uma cláusula `ORDER BY` e uma `GROUP BY` diferente, ou se a `ORDER BY` ou `GROUP BY` conterem colunas de tabelas diferentes da primeira tabela na fila de join, uma tabela temporária será criada.
- Se você utilizar `SQL_SMALL_RESULT`, o MySQL usará a tabela temporária em memória.
- Cada índice de tabela é consultado e o melhor índice que cobrir menos de 30% dos registros é usado. Se nenhum índice for encontrado, uma varredura rápida é feita pela tabela.
- Em alguns casos, o MySQL pode ler registros do índice mesmo sem consultar o arquivo de dados. Se todas colunas usadas do índice são numéricas, então somente a árvore de índice é usada para resolver a consulta.
- Antes de dar saída em cada registro, aqueles que não combinam com a cláusula `HAVING` são ignorados.

Some examples of queries that are very fast:

```
mysql> SELECT COUNT(*) FROM tbl_name;
mysql> SELECT MIN(key_part1),MAX(key_part1) FROM tbl_name;
mysql> SELECT MAX(key_part2) FROM tbl_name
    ->        WHERE key_part_1=constant;
mysql> SELECT ... FROM tbl_name
    ->        ORDER BY key_part1,key_part2,... LIMIT 10;
mysql> SELECT ... FROM tbl_name
    ->        ORDER BY key_part1 DESC,key_part2 DESC,... LIMIT 10;
```

As seguintes consultas são resolvidas utilizando somente a árvore de índices (assumindo que as colunas indexadas são numéricas):

```
mysql> SELECT key_part1,key_part2 FROM tbl_name WHERE key_part1=val;
mysql> SELECT COUNT(*) FROM tbl_name
    ->        WHERE key_part1=val1 AND key_part2=val2;
mysql> SELECT key_part2 FROM tbl_name GROUP BY key_part1;
```

As consultas a seguir utilizam indexação para recuperar os registros na ordem de classificação sem um passo de ordenação separado:

```
mysql> SELECT ... FROM tbl_name
    ->            ORDER BY key_part1,key_part2,... ;
mysql> SELECT ... FROM tbl_name
    ->            ORDER BY key_part1 DESC,key_part2 DESC,... ;
```

5.2.5 Como o MySQL Otimiza `IS NULL`

O MySQL pode fazer a mesma otimização em `column IS NULL` que ele pode com `column = constant_value`. Por exemplos, o MySQL pode usar índices e faixas para buscar por `NULL` com `IS NULL`.

```
SELECT * FROM table_name WHERE key_col IS NULL;

SELECT * FROM table_name WHERE key_col <=> NULL;

SELECT * FROM table_name WHERE key_col=# OR key_col=# OR key_col IS NULL
```

Se você usa `column_name IS NULL` em um `NOT NULL` em uma cláusula WHERE na tabela que não é usada no `OUTER JOIN`, esta espressão será otimizada de qualquer forma.

O MySQL 4.1. pode adicionalmente otimizar a combinação `column = expr AND column IS NULL`, uma forma que é comum em sub queries resolvidas. `EXPLAIN` mostrará `ref_or_null` quando esta otimização é usada.

Esta otimização pode tratar um `IS NULL` para qualquer parte da chave.

Alguns exemplos de consultas que são otimizadas (assumindo chave em t2 (a,b)):

```
SELECT * FROM t1 WHERE t1.a=expr OR t1.a IS NULL;

SELECT * FROM t1,t2 WHERE t1.a=t2.a OR t2.a IS NULL;

SELECT * FROM t1,t2 WHERE (t1.a=t2.a OR t2.a IS NULL) AND t2.b=t1.b;

SELECT * FROM t1,t2 WHERE t1.a=t2.a AND (t2.b=t1.b OR t2.b IS NULL);

SELECT * FROM t1,t2 WHERE (t1.a=t2.a AND t2.a IS NULL AND ...) OR (t1.a=t2.a AND t2
```

`ref_or_null` funciona fazendo primeiro uma leitura na chave indicada e depois disto uma busca separada por linhas com chave NULL.

Note que a otimização só pode tratar um nível `IS NULL`.

```
SELECT * FROM t1,t2 where (t1.a=t2.a AND t2.a IS NULL) OR (t1.b=t2.b AND t2.b IS NU
```

No caso acima o MySQL só usará busca de chave na parte `(t1.a=t2.a AND t2.a IS NULL)` e não poderá usar a parte da chave em `b`.

5.2.6 Como o MySQL Otimiza Cláusulas `DISTINCT`

`DISTINCT` combinado com `ORDER BY` também irá em vários casos criar uma tabela temporária.

Note que como `DISTINCT` pode usar `GROUP BY`, você deve estar ciente de como o MySQL funciona com campos na parte `ORDER BY` ou `HAVING` que não são parte dos campos selecionados. Veja Seção 6.3.7.3 [GROUP-BY-hidden-fields], Página 561.

Quando combinando `LIMIT row_count` com `DISTINCT`, o MySQL irá parar logo que encontrar `row_count` registros únicos.

Se você não utiliza colunas de todas tabelas usadas, o MySQL irá parar a varredura das tabelas não usadas logo que encontrar a primeira coincidência.

```
SELECT DISTINCT t1.a FROM t1,t2 where t1.a=t2.a;
```

Neste caso, assumindo que `t1` é usando antes de `t2` (confira com `EXPLAIN`), MySQL irá parar de ler de `t2` (para aquele registro particular em `t1`) quandoo primeiro registro em `t2` for encontrado.

5.2.7 Como o MySQL Otimiza `LEFT JOIN` e `RIGHT JOIN`

`A LEFT JOIN B join_condition` no MySQL está implementada como a seguir:

- A tabela `B` é configurada para ser dependente da tabela `A` e de todas as tabelas das quais `A` depende.
- A tabela `A` é configurada para ser dependente de todas as tabelas (exceto `B`) que são usadas na condição `LEFT JOIN`.
- A condição `LEFT JOIN` é usada para decidir como devemos recuperar registros a partir da tabela B. (Em outras palavras, qualquer condição na claúsula `WHERE` não é usada).
- Todas as otimizações padrões de join são feitas, com a excessão que uma tabela é sempre lida depois de todas as tabelas das quais é dependente. Se existir uma dependência circular o MySQL irá emitir um erro.
- Todas as otimizações padrões de `WHERE` são realizadas.
- Se existir um registro em `A` que coincida com a cláusula `WHERE`, mas não existir nenhum registro em `B` que coincida com a condição `ON` então um registro extra em `B` é gerado com todas as colunas com valor `NULL`.
- Se você utiliza `LEFT JOIN` para encontrar registros que não existem em alguma tabela e está usando o seguinte teste: `nome_coluna IS NULL` na parte `WHERE`, onde nome_colun é um campo que é declarado como `NOT NULL`, então o MySQL para de pesquisar por mais registros (para uma combinação particular de chaves) depois de ter encontrado um registro que combinar com a condição `LEFT JOIN`.

`RIGHT JOIN` é implementado de forma análoga à `LEFT JOIN`.

A ordem de leitura das tabelas forçada por `LEFT JOIN` e `STRAIGHT JOIN` irá ajudar o otimizador de joins (que calcula em qual ordem as tabelas devem ser unidas) a fazer seu trabalho mais rapidamente, já que haverão poucas permutações de tabelas a serem conferidas.

Perceba que o texto acima significa que se você fizer uma consulta do tipo:

```
SELECT * FROM b,a LEFT JOIN c ON (c.key=a.key) LEFT JOIN d (d.key=a.key)
        WHERE b.key=d.key
```

A partir do MySQL 4.0.14, o MySQL faz a seguinte otimização `LEFT JOIN`:

Se a condição `WHERE` é sempre falsa para a linha `NULL` gerada, o `LEFT JOIN` é alterado para um join normal.

Por exemplo, na seguinte consulta a cláusula `WHERE` seria falso se t2.coluna fosse `NULL`, asssim é seguro converter para uma join normal.

```
SELECT * FROM t1 LEFT t2 ON (column) WHERE t2.column2 =5;
->
SELECT * FROM t1,t2 WHERE t2.column2=5 AND t1.column=t2.column;
```

Isto pode ser feito mais rápido já que o MySQL pode agora usar a tabela `t2` antes da tabela `t1` se resultasse consulta melhor. Para forçar uma ordem de tabela específica, use `STRAIGHT JOIN`.

O MySQL irá fazer uma pesquisa completa em `b` já que o `LEFT JOIN` irá força-lo a ser lido antes de `d`.

A correção neste caso é alterar a consulta para:

```
SELECT * FROM b,a LEFT JOIN c ON (c.key=a.key) LEFT JOIN d (d.key=a.key)
         WHERE b.key=d.key
```

5.2.8 Como o MySQL Otimiza Cláusulas `ORDER BY`

Em alguns casos o MySQL pode utilizar índices para satisfazer uma requisição de `ORDER BY` ou `GROUP BY` sem fazer uma ordenação extra.

O índice também pode ser usado mesmo se o `ORDER BY` não coincidir exatamente com o índice, uma vez que todas as partes de índices não usadas e todos os extras na coluna `ORDER BY` são constantes na cláusula `WHERE`. A seguinte consulta usará o índice para resolver a parte `ORDER BY` / `GROUP BY`:

```
SELECT * FROM t1 ORDER BY key_part1,key_part2,...
SELECT * FROM t1 WHERE key_part1=constante ORDER BY key_part2
SELECT * FROM t1 WHERE key_part1=constante GROUP BY key_part2
SELECT * FROM t1 ORDER BY key_part1 DESC,key_part2 DESC
SELECT * FROM t1 WHERE key_part1=1 ORDER BY key_part1 DESC,key_part2 DESC
```

Alguns casos onde o MySQL **não** pode usar índices para resolver o `ORDER BY`: (Note que o MySQL ainda usará índices para encontrar o registro que coincide com a cláusula `WHERE`):

- Você está fazendo um `ORDER BY` em diferentes chaves:

  ```
  SELECT * FROM t1 ORDER BY key1,key2
  ```

- Você está fazendo um `ORDER BY` usando partes de chaves não consecutivas.

  ```
  SELECT * FROM t1 WHERE key2=constant ORDER BY key_part2
  ```

- Você está misturando `ASC` e `DESC`.

  ```
  SELECT * FROM t1 ORDER BY key_part1 DESC,key_part2 ASC
  ```

- As chaves usadas para buscar os registros são as mesmas usadas para fazer o `ORDER BY`:

  ```
  SELECT * FROM t1 WHERE key2=constant ORDER BY key1
  ```

- Você está unindo muitas tabelas e as colunas nas quais você está fazendo um `ORDER BY` não são todas da primeira tabela que não é `const` e que é usada para retornar registros. (Esta é a primeira tabela na saída do `EXPLAIN` que não usa um método de busca de registro `const`).
- Você tem diferentes expressões `ORDER BY` e `GROUP BY`.
- O índice da tabela usada é um tipo de índice que não armazena registros em ordem. (Como o índice `HASH` em tabelsn `HEAP`).

Nestes casos onde o MySQL tem que ordenar o resultado, ele usa o seguinte algoritmo:

- Lê todos os registros de acordo com a chave ou por uma varredura da tabela. Registros que não coincidem com a cláusula `WHERE` são saltados.

- Armazena a chave ordenada em um buffer (de tamanho `sort_buffer`).
- Quando o buffer ficar cheio, execute ordeno-o e armazene o resultado em um arquivo temposrário. Salve um ponteiro para o bloco ordenado. (No caso de todos os regitros caberem no buffer ordenado, nenhum arquivo temporário é criado).
- Repete o armazenamento acima até todas as linhas tenham sido lidos.
- Faz um multi-merge até `MERGEBUFF` (7) regiões para um bloco em outro arquivo temporário. Repete até que todos os blocos do primeiro arquivo estejam no segundo arquivo.
- Repete o seguinte até que restem menos que `MERGEBUFF2` (15) blocos.
- No último multi-merge, só o ponteiro para o registro (última parte de chave ordenada) é escrito em um arquivo de resultado.
- Agora o código em '`sql/records.cc`' será usado para ler através deles ordenadamente usando os ponteiros de registro no arquivo resultante. Para otimização , lemos em um grande bloco de ponteiros de registros, ordena-os então lemos o registros ordenadamente de de um buffer de registro. (`read_rnd_buffer_size`) .

Você pode verificar com `EXPLAIN SELECT ... ORDER BY` se o MySQL pode usar índices para resolver a consulta. Se você obtiver `Using filesort` na coluna `extra`, então o MySQL não pode usar índices para resolver o `ORDER BY`. Veja Seção 5.2.1 [EXPLAIN], Página 424.

Se você quiser ter uma velocidade `ORDER BY` maior, primeiro você deve ver se você pode fazer que o MySQL use índices em vez de fazer um fase de ordenação extra. Se não for possível, então você pode fazer:

- Aumente o tamanho da variável `sort_buffer_size`.
- Aumente o temenho da variável `read_rnd_buffer_size`.
- Altere `tmpdir` para apontar para um disco dedicado com muito espaço vazio. Se você usa o MySQL 4.1 ou posterior você pode distribuir a carga entre diversos discos físicos definindo `tmpdir` com uma lista de caminhos separados por dois pontos `:` (ponto e vírgula `;` no Windows). Eles serão usados de acordo com o método round-robin. **Nota:** Estes caminho devem estar em diferentes discos **físicos**, e não em diferentes partições do mesmo disco.

Por padrão, o MySQL ordena todas as consultas `GROUP BY x,y[,...]` como se você tivesse especificado `ORDER BY x,y[,...]`. Se você incluir a cláusula `ORDER BY` explicitamente, o MySQL a otimizará sem qualquer penalidade na velocidade, embora a ordenacao ainda ocorra. Se a consulta inclui um `GROUP BY` mas você deseja evitar a sobrecarga da ordenar o resultado, você pode suprimir a ordenacao especificando `ORDER BY NULL`:

```
INSERT INTO foo SELECT a,COUNT(*) FROM bar GROUP BY a ORDER BY NULL;
```

5.2.9 Como o MySQL Otimiza Cláusulas `LIMIT`

Em alguns casos o MySQL irá tratar a consulta de maneira diferente quando você estiver utilizando `LIMIT row_count` e não estiver utilizando `HAVING`:

- Se você estiver selecionando apenas alguns registros com `LIMIT`, o MySQL usará índices em alguns casos quando ele normalmente preferiria fazer uma varredura completa na tabela.

- Se você utilizar `LIMIT row_count` com `ORDER BY`, O MySQL irá terminar a ordenação logo que ele encontrar os primeiros `row_count` registros em vez de ordenar a tabela inteira.
- Ao combinar `LIMIT row_count` com `DISTINCT`, o MySQL irá parar logo que ele encontrar `row_count` registros únicos.
- Em alguns casos um `GROUP BY` pode ser resolvido lendo a chave em ordem (ou fazer uma classificação na chave) e então calcular resumos até o valor da chave alterar. Neste caso, `LIMIT row_count` não irá calcular nenhum `GROUP BY` desnecessário.
- Logo que o MySQL enviar os primeiros `#` registros para o cliente, ele irá abortar a consulta.
- `LIMIT 0` irá sempre retornar rapidamente um conjunto vazio. Isto é util para conferir a consulta e obter os tipos de campos do resultado.
- Quando o servidor utiliza tabelas temporárias para resolver a consulta, o `LIMIT row_count` é usado para calcular a quantidade de espaço necessário.

5.2.10 Performance das Consultas que Utilizam `INSERT`

O tempo para inserir um registro consiste aproximadamente de:

- Conexão: (3)
- Enviar a consulta para o servidor: (2)
- Analisar a consulta (2)
- Inserir o registro: (1 x tamanho do registro)
- Inserir os índices: (1 x número de índices)
- Fechar: (1)

onde os números são de certa forma proporcionais ao tempo total. Isto não leva em consideracão o sobrecarga inicial para abrir tabelas (que é feita uma vez para cada consulta concorrente em execução).

O tamanho da tabela diminuem a velocidade da inserção de índices em N log N (Arvores B).

Algumas maneiras de acelerar as inserções:

- Se você estiver inserindo vários registros do mesmo cliente ao mesmo tempo, utilize instruções `INSERT` com listas de múltiplos valores. Isto é muito mais rápido (muitas vezes em alguns casos) do que utilizar instruções `INSERT` separadas. Se você esta adicionando dados a uma tabela que não está vazia, você pode ajustar a variável `bulk_insert_buffer_size` para tornár isto mais rápido. Veja Seção 4.6.8.4 [`bulk_insert_buffer_size`], Página 309.
- Se você inserir vários registros de diferentes clientes, você pode obter velocidades mais altas utilizando a instrução `INSERT DELAYED`. Veja Seção 6.4.3 [INSERT], Página 579.
- Perceba que com `MyISAM` você pode inserir registros ao mesmo tempo que `SELECT`s estejam executando se não existirem registros apagados nas tabelas.
- Ao carregar uma tabela de um arquivo texto, utilize `LOAD DATA INFILE`. Isto é normalmente 20 vezes mais rápido do que utilizar várias instruções `INSERT` Veja Seção 6.4.8 [`LOAD DATA`], Página 588.

- É possível com algum trabalho extra fazer o `LOAD DATA INFILE` executar ainda mais rápido quando a tabela tiver vários índices. Utilize o seguinte procedimento:
  1. Opcionalmente crie a tabela com `CREATE TABLE`. Por exemplo, utilizando `mysql` ou Perl-DBI.
  2. Execute a instrução `FLUSH TABLES` ou o comando shell `mysqladmin flush-tables`.
  3. Utilize `myisamchk --keys-used=0 -rq /path/to/db/nome_tabela`. Isto removerá o uso de todos os índices da tabela.
  4. Insira dados na tabela com `LOAD DATA INFILE`. Isto não atualizará índices e será muito mais rápido.
  5. Se no futuro você precisar da tabela somente para leitura, execute `myisampack` na mesma para torná-la menor. Veja Seção 7.1.2.3 [Formato compactado], Página 635.
  6. Recrie os índices com `myisamchk -r -q /caminho/para/bd/nome_tabela`. Isto criará a árvore de índices em memória antes de escrevê-la para o disco, que é muito mais rápido porque evita que seja feita muita busca disco. A árvore de índices resultante é também balanceada perfeitamente.
  7. Execute uma instrução `FLUSH TABLES` ou o comando shell `mysqladmin flush-tables`.

  Note que `LOAD DATA INFILE` també faz a otimização acima se você a inserção for em uma tabela vazia; a principal diferença com o procedimento acima é qeu você pode deixar o `myisamchk` alocar muita mais memória temporária para a criação do índice que você deseje que o MySQL alocasse para todas as recriações de índice.

  Desde o MySQL 4.0 você também pode usar `ALTER TABLE nome_tbl DISABLE KEYS` em vez de `myisamchk --keys-used=0 -rq /caminho/para/bd/nome_tbl` e `ALTER TABLE nome_tbl ENABLE KEYS` em vez de `myisamchk -r -q /caminho/para/bd/nome_tbl`. Deste modo você também pode saltar os passos `FLUSH TABLES`.
- Você pode acelerar inserções feitas usando várias instruções bloqueando suas tabelas:

```
mysql> LOCK TABLES a WRITE;
mysql> INSERT INTO a VALUES (1,23),(2,34),(4,33);
mysql> INSERT INTO a VALUES (8,26),(6,29);
mysql> UNLOCK TABLES;
```

  A principal diferença na velocidade é que o buffer de índices é descarregado no disco somente uma vez, depois de todas instruções `INSERT` term sido completadas. Normalmente existiria tantas descargas do buffer de índices quanto instruções `INSERT` diferentes. O bloqueio não é necessário se você pode inserir todos registros com uma simples instrução.

  Para tabelas transacionais, você deve usar `BEGIN/COMMIT` em vez de `LOCK TABLES` para conseguir um aumento na velocidade.

  O bloqueio irá também diminuir o tempo total de testes de multi-conexões, mas o tempo máximo de espera para algumas threads irá aumentar (porque eles esperam pelos bloqueios). Por exemplo:

```
thread 1 faz 1000 inserções
thread 2, 3 e 4 faz 1 inserção
thread 5 faz 1000 inserções
```

Se você não estiver usando travas, 2, 3 e 4 irão terminar antes de 1 e 5, Se estiver utilizando travas, 2, 3 e 4 provavelmente não irão terminar antes de 1 ou 5, mas o tempo total deve ser cerca de 40% mais rápido.

Como as operações `INSERT`, `UPDATE` e `DELETE` são muito rápidas no MySQL, você obterá melhor perfomance geral adicionando travas em tudo que fizer mais que cerca de 5 inserções ou atualizações em um registro. Se você fizer várias inserções em um registro, você pode utilizar `LOCK TABLES` seguido de um `UNLOCK TABLES` de vez em quando (em torno de 1000 registro) para permitr que outras threads acessem a tabela. Isto também continua mostrando um bom ganho de performance.

Com certeza, `LOAD DATA INFILE` é muito mais rápido para carregar dados.

Para obter mais velocidade para `LOAD DATA INFILE` e `INSERT`, aumente o tamanho do buffer de chaves. Veja Seção 5.5.2 [Parâmetros de servidor], Página 454.

5.2.11 Performance das Consultas que Utilizam `UPDATE`

Consultas de atualização são otimizadas como uma consulta que usa `SELECT` com a sobrecarga adicional de escrita. A velocida da escrita depende do tamanho dos dados e do número de índices que serão atualizados. Índices que não forem alterados não serão atualizados.

Outra forma para obter atualizações rápidas é atrasar as atualizações e então fazer várias atualizações em um registro posteriormente. Fazer várias atualizações em um registro é muito mais rápido do que fazer uma por vez se você travar a tabela.

Perceba que, com formato de registros dinâmicos, atualizar um registro para um valor maior que o tamanho total pode dividir o registro. Portanto, se você faz isso frequentemente, é muito importante usar `OPTIMZE TABLE` de vez em quando. Veja Seção 4.6.1 [`OPTIMIZE TABLE`], Página 298.

5.2.12 Performance das Consultas que Utilizam `DELETE`

Se você deseja apagar todos os registros em uma tabela, deve usar `TRUNCATE TABLE nome_tabela`. Veja Seção 6.4.6 [`TRUNCATE`], Página 587.

O tempo para apagar um registro é exatamente proporcional ao número de índices. Para apagar registros mais rapidamente, você pode aumentar o tamanho do cache de índices. Veja Seção 5.5.2 [Parâmetros do servidor], Página 454.

5.2.13 Mais Dicas sobre Otimizações

Dicas não ordenadas para sistemas rápidos:

- Utilize conexões persistentes aos banco de dados para evitar a sobrecarga da conexão. Se você não poder utilizar conexões persistentes e for fazer várias novas conexões para o banco de dados, você pode desejar alterar o valor da variável `thread_cache_size`. Veja Seção 5.5.2 [Parâmetros do servidor], Página 454.
- Sempre verifique se todas as suas consultas realmente utilizam os índices que foram criados nas tabelas. No MySQL você pode fazer isto com o comando `EXPLAIN`. Veja Seção 5.2.1 [Explain], Página 424.

- Tente evitar consultas `SELECT` complexas em tabelas que são muito atualizadas. Isto evita problemas com travamento de tabelas.
- Com tabelas `MyISAM` que não tenham linhas deletadas, você pode inserir registros ao mesmo tempo que outra tabela a estiver lendo. Se este recurso é importante para você, deve considerar métodos onde você não tem que apagar registrou ou executar `OPTIMIZE TABLE` depois de ter apagado vários registros.
- Utilize `ALTER TABLE ... ORDER BY expr1,expr2...` se você na maioria das vezes recupera registros na ordem expr1,expr2... Utilizando esta opção depois de grandes alterações para a tabela, pode lhe dar um ganho de performance.
- Em alguns casos pode fazer sentido introduzir uma coluna 'hash' baseada nas informações das outras colunas. Se esta coluna for curta e razoavelmente única pode ser muito mais rápido do que ter um grande índice em várias colunas. No MySQL é muito fácil usar esta coluna extra: `SELECT * FROM nome_tabela WHERE hash=MD5(concat(col1,col2)) AND col_1='constante' AND col_2='constante'`
- Para tabelas que alteram muito você deve tentar evitar todas colunas `VARCHAR` ou `BLOB`. Você terá tamanho de registro dinâmico assim que usar um simples campo `VARCHAR` ou `BLOB`. Veja Cap“ptexi tulo 7 [Tipos de tabelas], Página 630.
- Normalmente não é muito útil cortar uma tabela em diferentes tabelas apenas porque os registros estão 'grandes'. Para acessar um registro, o maior problema para a performance é a busca em disco para encontra o primeiro byte do registro. Depois de encontrar os dados a maioria dos novos discos podem ler o registro inteiro rápido o bastante para a maioria das aplicações. Os únicos caos onde realmente faz sentido dividir uma tabela é se ela é uma tabela de registros com tamanho dinâmico (veja acima) que você pode alterar para um tamanho fixo, ou se você frequentemente precisa examinar a tabela e não precisa da maioria das colunas. Veja Cap“ptexi tulo 7 [Tipos de tabela], Página 630.
- Se frequentemente você precisar calcular alguma coisa baseada em informação de vários registros (ex: contagem de registros), provavlmente é melhor introduzir uma nova tabela e atualizar o contador em tempo real. Uma atualização do tipo `UPDATE table set count=count+1 where index_column=constante` é muito rapida!

  Isto é realmente importante quando você usa bancos de dados como o MySQL que só tem travamento de tabelas (multiplos leituras/escrita única). Isto também dará melhor performance com a maioria dos banco de dados, já que o gerenciador de bloqueio de registro terá menos a fazer neste caso.
- Se você precisar colerar estatisicas de tabelas maiores, utilize tabelas resumo em vez de buscar em toda a tabela. Manter os resumos deve ser mais rápido que tentar criar estatitíscas instantaneamente. É muito mais rápido criar novas tabelas através dos logs quando as coisas mudam (dependendo das descisões de negócio) que ter que alterar a aplicação em execução.
- Se possível, deve-se classificar relatórios como 'instantâneo' ou 'estatísticos' onde os dados necessários para relatórios estaiísticos são gerados apenas com base nas tabelas resumo que são geradas a partir dos dados atuais.
- Tire vantagem do fato de que a coluna tem valores padrões. Insira valores explicitamente apenas quando os valores a serem inseridos diferem do padrão. Isto reduz a analise que o MySQL precisa fazer e aumenta a velocidade de inserção.

- Em alguns casos é conveniente empacotar e armazenar os dados em um campo blob. Neste caso você deve adicionar algum código em sua aplicação para empacotar/desempacotar as coisas no campo blob, mas isto pode poupar vários acessos a algum estágio. Isto é prático quando você possui dados que não conformam com uma estrutura estática de tabela.
- Normalmente, você deve tentar manter todos dados não-redundantes (o que é chamado de 3a forma normal na teoria de bancos de dados), mas você não deve ter medo de duplicar alguns itens ou criar tabelas de resumo se você precisar delas para ganhar mais velocidade.
- Stored Procedures ou UDF (funções definidas pelo usuários) pode ser uma boa forma para obter mais performance. Neste caso você deve, entretanto, sempre ter uma maneira de fazer isso de outra maneira (mais lenta) se você utilizar algum banco de dados que não suporta isto.
- Você sempr pode ganhar velocidade fazendo cache de perguntas/respostas na sua aplicação e tentando fazer várias inserções/atualizações ao mesmo tempo. Se seu banco de dados suporta travamento de tabelas (como o MySQL e Oracle), isto deve ajudar a garantir que o cache de índices é descarregado somente uma vez depois de todas atualizações.
- Use `INSERT /*! DELAYED */` quando não precisar saber quando os dados são gravados. Isto melhora a velocidade porque vários registros podem ser gravados com uma simples escrita em disco.
- Use `INSERT /*! LOW_PRIORITY */` quando você desejar que suas consultas sejam mais importantes.
- Use `SELECT /*! HIGH_PRIORITY */` para obter consultas que ignoram a fila. Isto é, a consulta é feita mesmo se alguem estiver esperando para fazer uma escrita.
- Use a instrução `INSERT` multi-linhas para armazenar vários registros com um comando SQL (vários servidores SQL suportam isto).
- Use `LOAD DATA INFILE` para carregar volumes maiores de dados. Isto é mais rápido que as inserções normais e mais rápido até quando o `myisamchk` for integrado no `mysqld`.
- Use colunas `AUTO_INCREMENT` para garantir valores únicos.
- Use `OPTIMIZE TABLE` de vez em quando para evitar fragmentação quando estiver usando formatos de tabela dinâmica. Veja Seção 4.6.1 [`OPTIMIZE TABLE`], Página 298.
- Use tabelas `HEAP` para obter mais velocidade sempre que possível. Veja Capítulo 7 [Tipos de tabelas], Página 630.
- Quando estiver usando uma configuração de servidor Web normal, imagens devem ser armazenadas como arquivos. Isto é, armazene apenas uma referência para o arquivo no banco de dados. A principal razão para isto é que um servidor Web normal é muito melhor trabalhando com cache de arquivos do que com conteúdo de banco de dados. Portanto será muito mais fácil obter um sistema rápido se você utilizar arquivos.
- Use tabelas em memória para dados não-críticos que são acessados frequentemente (como informações sobre o último banner visto para usuários que não possuem cookies).
- Colunas com informações identicas em diferentes tabelas devem ser declaradas idênticas e ter nomes idênticos. No entanto, antes da versão 3.23, você pode obter ligações mais lentas.

Tente manter os nomes mais simples (use `nome` em vez de `nome_cliente` na tabela cliente). Para deixar seus nomes portáveis para outros servidores SQL você deve mantê-los menores que 18 caracteres.

- Se você realmente precisa de alta velocidade, você deve verificar as interfaces de baixo nível para armazenagem de dados que os diferentes servidores SQL suportam! Por exemplo, para acessar tabelas MySQL `MyISAM` diretamente, você pode obter um aumento de velocidade de 2-5 vezes comparado ao uso da interface SQL. Para conseguir essa façanha, os dados devem estar no mesmo servidor que sua aplicação, e normalmente devem ser acessados por apenas um processo (porque travamento de arquivos externo são muito lentos). Os problemas acima podem ser eliminados introduzindo comandos `MyISAM` de baixo nível no servidor MySQL (isto pode ser a maneira mais fácil para aumentar a performance). Tenha cuidado em projetar a interface com o banco de dados, ela deve ser bem facil para suportar estes tipos de otimizações.
- Em vários casos é mais rápido acessar dados de um banco de dados (utilizando uma conexão ativa) do que acessar um arquivo texto, apenas pelo fato do banco de dados ser mais compacto do que o arquivo texto (se você estiver utilizando dados numéricos), e isto irá envolver menos acessos à disco. Você também irá poupar código porque não será necessário analisar seus arquivos texto para encontrar limites de registros e campos.
- Você pode também usar replicação para conseguir ainda mais performance nas suas aplicações. Veja Seção 4.11 [Replicação], Página 378.
- Declarando uma tabela com `DELAY_KEY_WRITE=1` irá tornar a atualização de índices mais rápida, pois as mesmas não serão escritas em disco até o arquivo ser fechado. O lado ruim é que você deve executar `myisamchk` nestas tabelas antes de iniciar o `mysqld` para garantir que os dados estão corretos se o `mysqld` for finalizado no meio da execução. Como a informação de chave pode sempre ser gerada a partir dos dados, você não deve perder nada usando `DELAY_KEY_WRITE`.

5.3 Detalhes sobre Locks

5.3.1 Como o MySQL Trava as Tabelas

Você pode encontrar uma discussão sobre diferentes métodos de bloqueios no apêndice. Veja Seção E.4 [Métodos de bloqueio], Página 1087.

Todos os bloqueios no MySQL são livres de deadlock, exceto para tipos de tabela `InnoDB` e `BDB`. Isto é gerenciado sempre requisitando todos os bloqueios necessários de uma vez no começo de uma consulta e sempre bloqueando as tabelas na mesma ordem.

Tipos de tabela `InnoDB` automaticamente adquire seus locks de registro e os tipos de tabela `BDB` seus locks de páginas, durante o processamento das instruções SQL, e não no início da transação.

O método de bloqueio que o MySQL utiliza para `ESCRITA` funciona da seguinte forma:

- Se não existirem travas na tabela, coloque uma bloqueio de escrita na mesma.
- Caso contrário, coloca a requisição de trava na fila de bloqueios para escrita.

O método de bloqueio que o MySQL utilizado para `LEITURA` funciona da seguinte maneira:

- Se não existirem tarvas na tabela, coloca um bloqueio de leitura na mesma.
- Caso contrário, coloca a requisição de trava na fila de bloqueios para leitura.

Quando um bloqueio é liberado, a trava fica disponível para as threads na fila de bloqueios de escrita, e então para as threads na fila de bloqueios de leitura.

Isto significa que se você possui várias atualizações em uma tabela, instruções `SELECT` irão esperar até que não existam mais atualizações.

Para contornar este problema no caso onde você precisa fazer várias operações de `INSERT` e `SELECT` em uma tabela, você pode inserir registros em uma tabela temporária e atualizar a tabela real com os registros da tabela temporária de uma só vez.

Isto pode ser feito usando o código a seguir:

```
mysql> LOCK TABLES real_table WRITE, insert_table WRITE;
mysql> INSERT INTO real_table SELECT * FROM insert_table;
mysql> TRUNCATE TABLE insert_table;
mysql> UNLOCK TABLES;
```

Você pode utilizar as opções `LOW_PRIORITY` com `INSERT`, `UPDATE` ou `DELETE` ou `HIGH_PRIORITY` com `SELECT` se você desejar priorizar a recuperação em alguns casos específicos. Também podei-se iniciar o `mysqld` com `--low-priority-updates` para obter o mesmo comportamento.

Utilizar `SQL_BUFFER_RESULT` pode também tornar a criação de locks de tabelas mais curtos.Veja Seção 6.4.1 [SELECT], Página 562.

Você também pode alterar o código de bloqueioss no '`mysys/thr_lock.c`' para usar uma fila simples. Neste caso, bloqueios de escrita e leitura devem ter a mesma prioridade, o que pode ajudar em algumas aplicações.

5.3.2 Detalhes sobre Lock de Tabelas

O código de bloqueio de tabelas no MySQL é livre de deadlock.

O MySQL utiliza bloqueio de tabelas (no lugar de bloqueio de registros ou colnas) em todos os tipos de tabelas, exceto tabelas `BDB`, para obter uma alta velocidade nos bloqueios. Para grandes tabelas, bloqueio de tabelas é MUITO melhor que bloqueio de registros para a maioria das aplicações, mas existem, é claro, algumas desvantagens.

Para tabelas `BDB` e `InnoDB`, O MySQL só utiliza bloqueio de tabelas se você bloquear explicitamente a tabela com `LOCK TABLES` ou executar um comando quer irá modificar todos os registros na tabela, como `ALTER TABLE`. Para estes tipos de tabelas nós recomendamos a você não utilizar `LOCK TABLES`.

No MySQL versão 3.23.7 ou superior , você pode inserir registros em tabelas `MyISAM` ao mesmo tempo que outras threads estão lendo da mesma tabela. Perceba que atualmente isto funciona somente se não existirem buracos depois de registros apagados na tabela no momento que a inserção é feita. Quando todos os buracos forem preenchidos com novos dados, inserções concorrentes irão automaticamente ser habilitadas novamente.

O bloqueio de tabelas habilita várias threads para lerem de uma tabela ao mesmo tempo, mas se uma thread desejar escrever a uma tabela, ela primeiramente deve obter acesso

exclusivo. Durante a atualização, todas outras threads que desejarem acessar esta tabela em particular irão esperar até que a atualização acabe.

Como atualizações em tabelas normalmente são consideradas mais importantes que `SELECT`, todas as instruções que atualizam uma tabela tem maior prioridade que instruções que simplesmente recuperam informações. Isto deve garantir que atualizações não fiquem na fila por terem sido passadas várias consultas pesadas em uma tabela específica. (Você pode alterar isto utilizando LOW_PRIORITY com a instrução que faz a atualização ou `HIGH_PRIORITY` com a instrução `SELECT`.)

A partir do MySQL versão 3.23.7 pode-se utilizadar a variável `max_write_lock_count` para forçar o MySQL a fornecer temporariamente a todas as instruções `SELECT`, que esperam por uma tabela, uma prioridade mais alta depois de um número específico de inserções em uma tabela.

O bloqueio de tabela não é, no entanto, muito bom sobre os seguintes cenários:

- Um cliente emite uma `SELECT` que exige muito tempo para ser executada.
- Outro cliente então executa um `UPDATE` na tabela usada. Este cliente terá que esperar até que a `SELECT` seja terminada.
- Outro cliente executa outra instrução `SELECT` na mesma tabela. Como `UPDATE` tem maior prioridade que `SELECT`, esta `SELECT` irá esperar pelo término da `UPDATE`. Ela também irá esperar pelo término da primeira `SELECT`!
- Uma thread está esperando por algo do tipo `disco cheio`, caso em que todas as threads que desejam acessar a tabela com problema irão ser colocadas em estado de espera até que mais espaço em disco seja disponível.

Algumas soluções possíveis para este problema são:

- Tente deixar suas instruções `SELECT` sempre rápidas. Você pode ter que criar algumas tabelas de resumo para fazer isto.
- Inicie o `mysqld` com `--low-priority-updates`. Isto irá fornecer a todas instruções que atualizam (modificam) uma tabela prioridade menor que uma instrução `SELECT`. Neste caso a última instrução `SELECT` no cenário anterior deveria executar antes da instrução `INSERT`.

  Você pode fornecer a uma instrução `INSERT`, `UPDATE` ou `DELETE` específica menor prioridade com o atributo `LOW_PRIORITY`.
- Inicie o `mysqld` com um valor baixo para **max_write_lock_count** para fornecer bloqueios de `LEITURA` depois de um certo número de bloqueios de `ESCRITA`.
- Você pode especificar que todas as atualizações de uma thread específica deve ser feita utilizando prioridade baixa com o comando SQL: `SET SQL_LOW_PRIORITY_UPDATES=1`. Veja Seção 5.5.6 [`SET OPTION`], Página 460.
- Você pode especificar que uma `SELECT` específica é muito importante com o atributo `HIGH_PRIORITY`. Veja Seção 6.4.1 [`SELECT`], Página 562.
- Se você tiver problemas com `INSERT` combinado com `SELECT`, utilize as novas tabelas `MyISAM`, pois elas suportam `SELECT`s e `INSERT`s concorrentes.
- Se você utiliza principalmente instruções `INSERT` e `SELECT` misturadas, o atributo `DELAYED` no `INSERT` provavelmente irá resolver seus problemas. Veja Seção 6.4.3 [`INSERT`], Página 579.

- Se você tiver problemas com `SELECT` e `DELETE`, a opção `LIMIT` para `DELETE` pode ajudar. Veja Seção 6.4.5 [`DELETE`], Página 585.

5.4 Otimizando a Estrutura de Banco de Dados

5.4.1 Opções do Projeto

O MySQL mantem dados de registros e índices em arquivos separados. Vários (quase todos) bancos de dados misturam dados de registros e índice no mesmo arquivo. Nós acreditamos que a escolha do MySQL é melhor para uma ampla escala de sistemas modernos.

Outra forma de armazenar os dados de registros é manter a informação para cada coluna em uma área separada (exemplos são o SDBM e o Focus). Isto irá causar um ponto de performance para toda consulta que acessar mais de uma coluna. Como isto degrada rapidamente quando mais de uma coluna é acessada, acreditamos que este modelo não é bom para propósitos gerais de bancos de dados.

O caso mais comum é aquele em que o índice e dados são armazenados juntos (como no Oracle/Sybase). Neste caso você irá encontrar a informação do registro na folha da página de índice. A coisa boa com este layout é que ele, em vários casos, dependendo de como o índice é armazenado no cache, salva uma leitura de disco. As desvantagens deste layout são:

- A varredura da tabela é muito mais lenta porque você tem que ler os índices para encontrar os dados.
- Não podem ser usados apenas a tabela de índices para recuperar dados para uma consulta.
- Você perde muito espaço de armazenagem, já que que os índices devem ser duplicados nos nós (pois os registros não podem ser armazenados nos nós).
- Deleções irão degenerar a tabela depois de um tempo (já que os índices nos nós normalmente não são atualizados na deleção).
- É mais difícil fazer o cache somente dos dados de índices.

5.4.2 Deixando os Dados com o Menor Tamanho Possível

Uma das otimizações mais básicas é tentar manter seus dados (e índices) utilizando o menor espaço possível no disco (e em memória). Isto pode fornecer grandes melhorias porque a leitura de disco é mais rápida e normalmente menos memória principal será usada. A indexação também exige menos recursos se for feita em colunas menores.

O MySQL suporta vários diferentes tipos de tabelas e formatos de registros. Você pode ter um ótimo ganho de performance escolhendo o formato certo de tabela a ser usada. Veja Cap“ptexi tulo 7 [Tipos de tabelas], Página 630.

Pode-se obter melhor performance em uma tabela e minimizar espaço de armazenagem utilizando as técnicas listadas abaixo:

- Utilize os tipos mais eficientes (menores) sempre que possível. O MySQL tem vários tipos especializados que economizam espaço em disco e memória.

- Utilize tipos inteiros menores se possível para obter tabelas menores. Por exemplo, `MEDIUMINT` normalmente é melhor que `INT`.
- Declare colunas para serem `NOT NULL` se possível. Isto deixa tudo mais rápido e você economiza um bit por coluna. Perceba que se você realmente precisa de `NULL` nas suas aplicações, podem ser usados. Tente simplesmente não usá-la em todas as colunas por padrão.
- Se você não possui nenhuma coluna de tamanho variável (`VARCHAR`, `TEXT` ou `BLOB`), um formato de registro de tamanho fixo para é utilizado. Isto é mais rápido mas infelizmente pode ocupar mais espaço. Veja Seção 7.1.2 [Formatos de tabelas MyISAM], Página 634.
- O índice primário de uma tabela deve ser o mais curto possível. Isto torna a identificação de um registro fácil e eficiente.
- Para cada tabela, você deve decidir qual metódo de armazenamento/índice utilizar. Veja Cap“ptexi tulo 7 [Tipos de tabelas], Página 630.
- Crie somente os índices necessários. Índices são bons para recuperação mas ruins quando você precisa armazenar os dados rapidamente. Se na maioria das vezes você acessa uma tabela pesquisando em uma combinação de colunas, crie um índice para elas. A primeira parte do índice deve ser a coluna mais utilizada. Se você SEMPRE utiliza várias colunas, deve usar a coluna com mais duplicações em primeiro lugar para obter melhor compactação do índice.
- Se for melhor que uma coluna tenha um prefixo único nos primeiros caracteres, é melhor indexar somente este prefixo. O MySQL suporta um índice em uma parte de uma coluna de caracteres. Índices menores são mais rápidos não somente porque eles exigem menos espaço em disco mas também porque eles irão fornecer a você mais acerto no cache de índice e isto diminui acessos a disco. Veja Seção 5.5.2 [Parâmetros de servidor], Página 454.
- Em algumas circunstâncias pode ser benéfico dividir uma tabela que é varrida frequentemente em duas. Isto é verdade especificamente se a tabela tiver um formato dinâmico e for possível utilizar um formato de tabela estático que possa ser usada para encontrar os registros relevantes quando se fizer uma varredura da tabela.

5.4.3 Como o MySQL Utiliza Índices

Os índices são utilizados para encontrar registros com um valor específico de uma coluna rapidamente. Sem um índice o MySQL tem de iniciar com o primeiro registro e depois ler através de toda a tabela até que ele encontre os registros relevantes. Quanto maior a tabela, maior será o custo. Se a tabela possui um índice para as colunas em questão, o MySQL pode rapidamente obter uma posição para procurar no meio do arquivo de dados sem ter que varrer todos os registros. Se uma tabela possui 1000 registros, isto é pelo menos 100 vezes mais rápido do que ler todos os registros sequencialmente. Note que se você precisar acessar quase todos os 1000 registros, seria mais rápido acessá-los sequencialmente porque evitaria acessos ao disco.

Todos os índices do MySQL (`PRIMARY`, `UNIQUE` e `INDEX`) são armazenados em árvores B. Strings são automaticamente compactadas nos espaços finais e prefixados. Veja Seção 6.5.7 [`CREATE INDEX`], Página 613.

Índices são utilizados nos seguintes modos:

- Para encontrar rapidamente os registros que coincidam com uma cláusula `WHERE`.
- Para recuperar registros de outras tabelas ao realizar joins.
- Para encontrar o valor `MAX()` ou `MIN()` para uma coluna indexada espeifica. Isto é otimizado por um preprocessador que confere se você está utilizando `WHERE` key_part_#=constante em todas as partes da chave `<` N. Neste caso o MySQL irá fazer uma simples procura na chave e trocar a expressão `MIN()` com uma constante. Se todas as expressões forem trocadas por constantes, a consulta retornará imediatamente:

  ```
  SELECT MIN(key_part2),MAX(key_part2) FROM nome_tabela where key_part1=10
  ```

- Para ordenar ou agrupar uma tabela se a ordenação ou agrupamento for feito em um prefixo mais à esquerda de uma chave util (por exemplo, `ORDER BY key_part_1, key_part_2`). A chave é lida na ordem invertida se todas as partes da chave forem seguidas por `DESC`. Veja Seção 5.2.8 [Otimização ORDER BY], Página 436.
- Em alguns casos uma consulta pode ser otimizada para recuperar valores sem consultar o arquivo de dados. Se todas colunas utilizadas para alguma tabela são numéricas e formam um prefixo mais à esquerda para alguma chave, os valores podem ser recuperados da árvore de índices para aumentar a velocidade:

  ```
  SELECT key_part3 FROM nome_tabela WHERE key_part1=1
  ```

Suponha que você utilize a seguinte instrução `SELECT`:

```
mysql> SELECT * FROM nome_tabela WHERE col1=val1 AND col2=val2;
```

Se um índice de colunas múltiplas existir em `col1` e `col2`, os registros apropriados podem ser recuperados diretamente. Se índices separados de únicas colunas existirem em `col1` e `col2`, o otimizador tentará encontrar o índice mais restritivo decidindo qual índice irá encontrar menos registros e usará este índice para recuperar os registros.

Se a tabela possuir um índice de múltiplas colunas, qualquer prefixo mais à esquerda do índice pode ser usado pelo otimizador para encontrar registros. Por exemplo, se você possui um índice de três colunas em `(col1, col2, col3)`, você tem capacidades de busca indexada em `(col1)`, `(col1, col2)` e `(col1, col2, col3)`.

O MySQL não pode utilizar um índice parcial se as colunas não formarem um prefixo mais à esquerda do índice. Suponha que você tenha as instruções `SELECT` mostradas abaixo:

```
mysql> SELECT * FROM nome_tabela WHERE col1=val1;
mysql> SELECT * FROM nome_tabela WHERE col2=val2;
mysql> SELECT * FROM nome_tabela WHERE col2=val2 AND col3=val3;
```

Se um índice existir em `(col1, col2, col3)`, somente a primeira consulta anteriores utiliza o índice. A segunda e terceira consultas involvem colunas indexadas, mas `(col2)` e `(col2, col3)` não são os prefixos mais à esquerda de `(col1, col2, col3)`.

O MySQL também utiliza índices para comparações do tipo `LIKE` se o argumento para `LIKE` for uma string constante que não inicie com um meta caracter Por exemplo as seguintes instruções `SELECT` utilizam índices:

```
mysql> SELECT * FROM nome_tbl WHERE key_col LIKE "Patrick%";
mysql> SELECT * FROM nome_tbl WHERE key_col LIKE "Pat%_ck%";
```

Na primeira instrução, somente os registros com `"Patrick" <= key_col < "Patricl"` são considerados. Na segunda instrução, somente registros com `"Pat" <= key_col < "Pau"` são considerados.

As seguintes instruções `SELECT` não usarão índices:

```
mysql> SELECT * FROM nome_tbl WHERE key_col LIKE "%Patrick%";
mysql> SELECT * FROM nome_tbl WHERE key_col LIKE other_col;
```

Na primeira instrução, o valor `LIKE` inicia com um meta caracter. Na segunda instrução, o valor `LIKE` não é uma constante.

O MySQL 4.0 faz outra otimização em `LIKE`. Se você usar `... LIKE "%string%"` e `string` tiver mais de 3 caracteres, o MySQL usará o algorítmo `Turbo Boyer-Moore` para inicializar o padrão para a string e então usar este padrão para realizar a pesquisa mais rápido.

Buscas usando `nome_coluna IS NULL` usa índices se `nome_coluna` é um índice.

O MySQL normalmente utiliza o índice que encontra o menor número de registros. Um índice é usado para colunas que você compara com os seguintes operadores: `=`, `>`, `>=`, `<`, `<=`, `BETWEEN` ou um `LIKE` com um padrão que começa com um prefixo sem meta caracteres como `'algo%'`.

Qualquer índice que não cobrem todos os níveis de `AND` na cláusula `WHERE` não é utilizado para otimizar a consulta. Em outras palavras: Para poder usar um índice, um prefixo do índice deve ser utilizado em todo agrupamento `AND`.

A seguinte cláusula `WHERE` utilizará índices:

```
... WHERE index_part1=1 AND index_part2=2 AND other_column=3
... WHERE index=1 OR A=10 AND index=2      /* index = 1 OR index = 2 */
... WHERE index_part1='hello' AND index_part_3=5
          /* optimised like "index_part1='hello'" */
... WHERE index1=1 AND index2=2 OR index1=3 AND index3=3;
          /* Can use index on index1 but not on index2 or index 3 */
```

Estas cláusulas `WHERE` **não** utilizam índices:

```
... WHERE index_part2=1 AND index_part3=2  /* index_part_1 is not used */
... WHERE index=1 OR A=10                  /* Index is not used in
                                                  both AND parts */
... WHERE index_part1=1 OR index_part2=10  /* No index spans all rows  */
```

Perceba que algumas vezes o MySQL não utilizará um índice, mesmo se algum estiver disponível. Um exemplo deste caso é quando o uso do índice necessita que o MySQL acesse mais de 30% dos registros na tabela. (Neste caso uma varredura da tabela é provavelmente mais rápido, já que ela necessitará de menos pesquisas em discos). No entanto, se uma consulta utiliza `LIMIT` para recuperar somente parte dos registros, o MySQL irá utilizar um índice de qualquer forma, pois assim pode encontrar os poucos registros mais rapidamente e retornar o resultado.

5.4.4 Índices de Colunas

Todos os tipos de colunas do MySQL podem ser indexadas. O uso de índices nas colunas relevantes é a melhor forma de melhorar a performance de operações `SELECT`.

O número máximo de índices por tabelas e o tamanho máximo de um índice é definido pelo mecanismo de armazenamento. Veja Cap“ptexi tulo 7 [Table types], Página 630. Todos os mecanismos de armazenamentos suportam um mínimo de 16 chaves por tabela e um índice de tamanho total mínimo de 256 bytes.

Para colunas `CHAR` e `VARCHAR` você pode indexar um prefixo da coluna. Isto é muito mais rápido e necessita de menos espaço em disco do que indexar a coluna inteira. A sintaxe para utilizar na instrução `CREATE TABLE` para indexar um prefixo de uma coluna se parece com o exemplo a seguir:

```
INDEX nome_indice (nome_campo(tamanho))
```

O exemplo abaixo cria um índice para os primeiros 10 caracteres da coluna `nome`:

```
mysql> CREATE TABLE teste (
           nome CHAR(200) NOT NULL,
           INDEX nome_indice (nome(10)));
```

Para colunas `BLOB` e `TEXT`, você deve indexar um prefixo da coluna. O índice pode ter até 255 bytes.

No MySQL Versão 3.23.23 ou posterior, você pode também criar índices **FULLTEXT** especiais. Eles são utilizados para pesquisas textuais. Somente o tipo de tabela `MyISAM` suporta índices `FULLTEXT` e apenas para colunas `CHAR`, `VARCHAR` e `TEXT`. Indexação sempre acontece sobre toda a coluna e indexação parcial (prefixo) não é suportada. Veja Seção 6.8 [Fulltext Search], Página 619 para detalhes.

5.4.5 Índices de Múltiplas Colunas

O MySQL pode criar índices em múltiplas colunas. Um índice pode consistir de até 15 colunas. (Em colunas `CHAR` e `VARCHAR` você também pode utilizar um prefixo da coluna como parte de um índice).

Um índice de múltiplas colunas pode ser considerado um array ordenado contendo valores que são criados concatenando valores de colunas indexadas.

O MySQL utiliza índices de múltiplas colunas de forma que consultas são rápidas quando você especifica uma quantidade conhecida para a primeira coluna do índice em uma cláusula `WHERE`, mesmo se você não especificar valores para as outras colunas.

Suponha que uma tabela tenha a seguinte especificação:

```
mysql> CREATE TABLE teste (
           id INT NOT NULL,
           ultimo_nome CHAR(30) NOT NULL,
           primeiro_nome CHAR(30) NOT NULL,
           PRIMARY KEY (id),
           INDEX nome (ultimo_nome,primeiro_nome));
```

Então o índice `nome` é um índice com `ultimo_nome` e `primeiro_nome`. O índice será usado para consultas que especificarem valores em um limite conhecido para `ultimo_nome`, ou para ambos `ultimo_nome` e `primeiro_nome`. Desta forma, o índice `nome` será usado nas seguintes consultas:

```
mysql> SELECT * FROM teste WHERE ultimo_nome="Widenius";
```

```
mysql> SELECT * FROM teste WHERE ultimo_nome="Widenius"
                           AND primeiro_nome="Michael";

mysql> SELECT * FROM teste WHERE ultimo_nome="Widenius"
                          AND (primeiro_nome="Michael" OR primeiro_nome="Monty");

mysql> SELECT * FROM teste WHERE ultimo_nome="Widenius"
                          AND primeiro_nome >="M" AND primeiro_nome < "N";
```

Entretanto, o índice `nome` **não** será usado nas seguintes consultas:

```
mysql> SELECT * FROM teste WHERE primeiro_nome="Michael";

mysql> SELECT * FROM teste WHERE ultimo_nome="Widenius"
                          OR primeiro_nome="Michael";
```

Para maiores informações sobre a maneira que o MySQL utiliza índices para melhorar o desempenho das consultas, veja Seção 5.4.3 [índices do MySQL], Página 447.

5.4.6 Como o MySQL Conta as Tabelas Abertas

Ao executar o comando `mysqladmin status`, você verá algo deste tipo:

```
Uptime: 426 Running threads: 1 Questions: 11082 Reloads: 1 Open tables: 12
```

O valor `Open tables` de 12 ode ser bastante estranho se você só possui 6 tabelas.

O MySQL é multithreaded, portanto ele pode haver clientes enviando consultas para uma determinada tabela simultaneamente. Para minimizar o problema com dois clientes tendo diferentes estados no mesmo arquivo, a tabela é aberta independentemente por cada thread concorrente. Isto exige mais memória mas normalmente aumentará o desempenho. Com tabelas `ISAM` e `MyISAM`, um descritor extra de arquivo é necessário para o arquivo de dados, para cada cliente que tem a tabela aberta. O descritor de arquivo de índice é compartilhado entre todas as threads.

Você pode ler mais sobre este tópico na próxima seção. Veja Seção 5.4.7 [Cache de tabelas], Página 451.

5.4.7 Como o MySQL Abre e Fecha as Tabelas

As variáveis do servidor `table_cache`, `max_connections` e `max_tmp_tables` afetam o número máximo de arquivos que o servidor mantêm abertos. Se você aumentar um ou ambos destes valores, você pode ir contra um limite imposto pelo seu sistema operacional no número de arquivos abertos por processo. Você pode aumentar o limite de arquivos abertos em muitos sistemas operacionais, embora o método varia muito de um sistema para outro. Consulte a documentação de seu Sistema Operacional para saber como fazê-lo, porque o método para alterar o limite varia muito de um sistema para outro.

`table_cache` é relacionado a `max_connections`. Por exemplo, para 200 conexões concorrentes em execução, você deve ter um tamanho de cache de tabela de pelo menos `200 * n`, onde `n` é o número máximo de tabelas em um join. Você também precisa reservar alguns descritores de arquivos para tabelas e arquivos temporários.

Esteja certo de que o seu sistema operacional pode tratar o número de descritores de arquivos abertos definido pelo valor de `table_cache`. Se `table_cache` for muito alto, o MySQL pode esgotar os descritores de arquivo e recusar conexões, falhar na execução de consultas e ser muito instavel. Você também têm que levar em conta que o mecanismo de armazenamento `MyISAM` precisa de dois descritores de arquivos para cada tabela aberta. Você pode aumentar o número de descritores de arquivo disponíveis para o MySQL com a opção de inicialização `--open-files-limit=#`. Veja Seção A.2.17 [Tratamento de arquivos insuficientes], Página 921.

A cache de tabelas abertas será mantido em um nível de `table_cache` entradas. O valor padrão é 64; isto pode ser alterado com a opção `-O table_cache=#` do `mysqld`. Note que o MySQL pode temporariamente abrir mais tabelas para poder se executar consultas.

Um tabela não usada é fechada e removida da cache de tabelas sob as seguintes circuntâncias:

- Quando a cache está cheia e um thread tenta abrir uma tabela que não está na cache.
- Quando a cache contém mais que `table_cache` entradas e uma thread não está mais usando uma tabela.
- Quando alguém executa `mysqladmin refresh` ou `mysqladmin flush-tables`.
- Quando alguém executa uma instrução `FLUSH TABLES`.

Quando o cache de tabela encher, o servidor usa o seguinte procedimento para encontrar uma entrada de cache para usar:

- Tabelas que não estiverem em uso são liberadas, na ordem LRU (least-recently-used), ou seja, a tabela que foi usada menos rcentemente.
- Se o cache estiver cheio e nenhuma tabelas pode ser liberada, mas uma nova tabela precisar ser aberta, o cache é extendido temporariamente quando necessário.
- Se o cache estiver no estado temporariamente extendido e uma tabela vai do estado em-uso para o fora-de-uso, a tabela é fechada e liberada do cache.

A table is opened for each concurrent access. This means the table needs to be opened twice if two threads access the same table or if a thread accesses the table twice in the same query (for example, by joining the table to itself).

Uma tabela é aberta para cada acesso simultâneo. Isto significa a tabela precisa ser aberta duas vezes se duas threads acessam a mesma tabela ou se uma thread acessa a tabela duas vezes na mesma consulta (por exemplo, fazendo um join da tabela com ela mesma). A primeira abertura de qualquer tabela exige dois descritores de arquivos; cada uso adicional da tabela exige somente um descritor. O descritor extra para a primeira abertura é para o arquivo de índice: este descritor é compartilhado entre todas as threads.

Se você está abrindo uma tabela com a instrução `HANDLER nome_tabela OPEN`, uma tabela dedicada é alocada para a thread. Este objeto da tabela não é compartilhado por outras threads e não será fechado até que a thread chame `HANDLER nome_tabela CLOSE` ou seja finalizada. Veja Seção 6.4.9 [HANDLER], Página 595. Quando isto acontece, a tabela é colocada de volta na cache de tabela (se a cache não estiver cheia).

Você pode conferir se o seu cache de tabela está muito pequeno conferindo a variável `opened_tables` do `mysqld`. Se este valor for muito grande, mesmo se você não fez vários `FLUSH TABLES`, você deve aumentar o tamanho da sua cache de tabelas. Veja Seção 4.6.8.3 [Opened_tables], Página 305.

5.4.8 Desvantagem em Criar um Número Grande de Tabelas no Mesmo Banco de Dados

Se você possui muitos arquivos em um diretório, operações de abrir, fechar e criação ficarão lentos. Se você executar instruções `SELECT` em diversas tabelas, existirá uma pequena sobrecarga quando o cache de tabela estiver cheio, porque para toda tabela que teve que ser aberta, outra deve ser fechada. Você pode reduzir esta sobrecarga tornando o cache de tabelas maior.

5.5 Otimizando o Servidor MySQL

5.5.1 Sintonia dos Parâmetros em Tempo de Sistema/Compilação e na Inicialização

Nós iniciamos com o fator do nível do sistema pois algumas destas decisões devem ser feitas bem cedo. Em outros casos uma rápida olhada para esta seção pode satisfazer porque ela não é tão importante para os grandes ganhos. Entretanto, é sempre bom ter ter noções de como você pode obter melhorias alterando coisas neste nível.

Qual sistema operacional a usar é realmente importante! Para obter o melhor uso de máquinas com múltiplas CPUs você deve utilizar Solaris (porque a sua implemetação das threads funcionam muito bem) ou Linux (porque o kernel 2.2 tem suporte SMP muito bom). Também, em Linux mais antigos temos o limite de tamanho de arquivo de 2G por padrão. Se você tem tal kernel e precisa desesperadamente de trabalhar com arquivos maiores que 2G em máquinas intel Linux, você deve obter o patch LFS para o sistema de arquivos ext2. Outros sistemas de arquivo como ReiserFS e XFS não possuem esta limitação de 2G.

Como ainda não temos o MySQL em produção em muitas outras plataformas, nós aconselhamos que você teste a plataforma pretendida antes de escolhe-la, se possível.

Outras dicas:

- Se você possui RAM suficiente, você pode remover todos os dispositivos de troca. Alguns sistemas operacionais irão utilizar um disposotico de troca em alguns contextos, mesmo se você possuir memória livre.
- Utilize a opção do MySQL `--skip-external-locking` para evitar locks externos. Perceba que isto não irá afetar a funcionalidade do MySQL se você estiver executando um único servidor. Apenas lembre-se de desligar o servidor (ou travar as partes relevantes) antes de executar `myisamchk`. Em alguns sistemas esta opção é obrigatório porque o lock externo não funcionam em nenhum caso.

  A opção `--skip-external-locking` está ligada por padrão a partir do MySQL 4.0. Antes disto, era ligada por padrão quando compilando com MIT-pthreads, porque `flock()` não é totalmente suportado pelas MIT-pthreads em todas plataformas. É também o padrão para Linux pois o bloqueio de arquivos no Linux não é muito seguro.

  O único caso que você não pode utilizar `--skip-external-locking` é se você precisa de vários *servidores* MySQL (não clientes) acessando os mesmos dados, ou executar `myisamchk` na tabela sem dizer ao servidor para descarregar e travar as tabelas primeiro

  Você pode continuar usando `LOCK TABLES`/`UNLOCK TABLES` mesmo se você estiver utilizando `--skip-external-locking`.

5.5.2 Parâmetros de Sintonia do Servidor

Você pode determinar tamanho padrão do buffer usados pelo servidor `mysqld` com este comando:

```
shell> mysqld --help
```

Este comando produz uma lista de todas as opções do `mysqld` e variáveis configuráveis. A saída inclui os valores padrão das variáveis e se parece com isto:

```
back_log                current value: 5
bdb_cache_size          current value: 1048540
binlog_cache_size       current value: 32768
connect_timeout         current value: 5
delayed_insert_timeout  current value: 300
delayed_insert_limit    current value: 100
delayed_queue_size      current value: 1000
flush_time              current value: 0
interactive_timeout     current value: 28800
join_buffer_size        current value: 131072
key_buffer_size         current value: 1048540
lower_case_nome_tabelas  current value: 0
long_query_time         current value: 10
max_allowed_packet      current value: 1048576
max_binlog_cache_size   current value: 4294967295
max_connections         current value: 100
max_connect_errors      current value: 10
max_delayed_threads     current value: 20
max_heap_table_size     current value: 16777216
max_join_size           current value: 4294967295
max_sort_length         current value: 1024
max_tmp_tables          current value: 32
max_write_lock_count    current value: 4294967295
myisam_sort_buffer_size current value: 8388608
net_buffer_length       current value: 16384
net_retry_count         current value: 10
net_read_timeout        current value: 30
net_write_timeout       current value: 60
read_buffer_size        current value: 131072
record_rnd_buffer_size  current value: 262144
slow_launch_time        current value: 2
sort_buffer             current value: 2097116
table_cache             current value: 64
thread_concurrency      current value: 10
tmp_table_size          current value: 1048576
thread_stack            current value: 131072
wait_timeout            current value: 28800
```

Se existir um servidor `mysqld` em execução, você pode ver quais valores ele está usando atualmente para as variáveis executando esta instrução:

```
mysql> SHOW VARIABLES;
```

Você também pode ver algumas estatísticas e indicadores de status para um servidor em execução executando este comando:

```
mysql> SHOW STATUS;
```

Para encontrar uma descrição completa de todas as variáveis na seção `SHOW VARIABLES` neste manual. Veja Seção 4.6.8.4 [`SHOW VARIABLES`], Página 309.

Para informação sobre variáveis de estado, veja Seção 4.6.8.3 [`SHOW STATUS`], Página 305.

Variáveis de servidor e informação de status também pode ser obtido usando `mysqladmin`:

```
shell> mysqladmin variables
shell> mysqladmin extended-status
```

O MySQL utiliza algorítmos que são muito escaláveis, portanto, normalmente você pode trabalhar com pouca memória. Entretanto, se você fornecer ao MySQL mais memória, obterá um desempenho melhor.

Quando estiver ajustando um servidor MySQL, as duas variáveis mais importantes que devem ser usadas são `key_buffer_size` e `table_cache`. Você deve se sentir confiante que as duas estejam corretas antes de tentar alterar qualquer outra variável.

Os seguintes exemplos indicam alguns valores típicos de variáveis para diferentes configurações de tempo de execução. Os exemplos usam o script `mysqld_safe` e usam a sintaxe `--name=value` para definir a variável `name` com o valor `value`. Esta sintaxe está disponível a partir do MySQL 4.0. Para versões mais antigas do MySQL, tome as seguintes diferenças nas contas:

- Use `safe_mysqld` em vez de `mysqld_safe`.
- Configure as variáveis usando a sintaxe `--set-variable=name=value` ou `-O name=value`
- Para nomes de variáveis que finalizam em `_size`, você pode precisar especificá-las sem `_size`. Por exemplo, o nome antigo para `sort_buffer_size` é `sort_buffer`. O nome antigo para `read_buffer_size` é `record_buffer`. Para ver quais variáveis a versão do seu servidor reconhece, use `mysqld --help`.

Se você possui pelo menos 256M de memória e várias tabelas e deseja obter o melhor desempenho com um número moderado de clientes, deve utilizar algo como:

```
shell> mysqld_safe --key_buffer_size=64M --table_cache=256 \
           --sort_buffer_size=4M --read_buffer_size=1M &
```

Se possui apenas 128M de memória e apenas algumas poucas tabelas, mas ainda deseja realizar várias ordenações, você pode utilizar:

```
shell> mysqld_safe --key_buffer_size=16M --sort_buffer_size=1M
```

Se você possuir pouca memória e tiver muitas conexões, utilize algo como:

```
shell> mysqld_safe --key_buffer_size=512K --sort_buffer_size=100K \
           --read_buffer_size=100K &
```

ou mesmo isto:

```
shell> mysqld_safe --key_buffer_size=512K --sort_buffer_size=16K \
           --table_cache=32 --read_buffer_size=8K -O net_buffer_length=1K &
```

Se você estiver executando um `GROUP BY` ou `ORDER BY` em tabelas que são muito maiores que sua memória disponível você deve aumentar o valor de `record_rnd_buffer_size` para acelerar a leitura de registros após a operação de ordenação.

Quando você tiver instalado o MySQL, o diretório '`support-files`' irá conter alguns arquivos exemplos do `my.cnf`, '`my-huge.cnf`', '`my-large.cnf`', '`my-medium.cnf`' e '`my-small.cnf`', você pode usá-los como base para otimizar seu sistema.

Se você possui várias conexões simultâneas, "problemas de trocas" podem ocorrer a menos que o `mysqld` tenha sido configurado para usar muito pouca memória para cada conexão. O `mysqld` tem melhor performance se você tiver memória suficiente para todas as conexões, é claro.

Perceba que se você especifica uma opção na linha de comando para o `mysqld`, ou `mysqld_safe` ele permanece em efeito somente para aquela chamada do servidor. Para usar a opção toda vez que o servidor executa, coloque-o em um arquivo de opção.

Para ver os efeitos de uma alteração de parâmetro, faça algo como:

```
shell> mysqld --key_buffer_size=32m --help
```

Tenha certeza que a opção `--help` seja a última do comando; de outra forma o efeito de qualquer opções listadas depois na linha de comando não serão refletidas na saída.

5.5.3 Como a Compilação e a Ligação Afetam a Velocidade do MySQL

A maioria dos testes seguintes são feitos no Linux com os benchmarks do MySQL, mas eles devem fornecer alguma indicação para outros sistemas operacionais e workloads.

Você obtêm um executável mais veloz quando ligado com `-static`.

No Linux, você irá obter o código mais rápido quando compilando com `pgcc` e `-O3`. Para compilar '`sql_yacc.cc`' com estas opções, você precisa de cerca de 200M de memória porque o `gcc/pgcc` precisa de muita memória para criar todas as funções em linha. Também deve ser configurado o parâmetro `CXX=gcc` para evitar que a biblioteca `libstdc++` seja incluida (não é necessária). Perceba que com algumas versões do `pgcc`, o código resultante irá executar somente em verdadeiros processadores Pentium, mesmo que você utilize a opção do compilador para o código resultante que você quer, funcionando em todos os processadores do tipo x586 (como AMD).

Só pelo fato de utilizar um melhor compilador e/ou melhores opções do compilador você pode obter um aumento de desempenho de 10-30% na sua aplicação. Isto é particularmente importante se você mesmo compila o servidor SQL!

Nós testamos ambos os compiladores Cygnus Codefusion e o Fujitsu, mas quando os testamos, nenhum dos dois era suficientemente livre de erros para que o MySQL compilasse com as otimizações.

Quando você compila o MySQL deve incluir suporte somente para os conjuntos de caracteres que deseja usar. (Opção `--with-charset=xxx`). As distribuições binárias padrão do MySQL são compiladas com suporte para todos os conjuntos de caracteres.

Segue uma lista de algumas medidas que temos feito:

- Se você utiliza o `pgcc` e compila tudo com `-O6`, o servidor `mysqld` é 1% mais rápido do que com o `gcc` 2.95.2.

- Se você liga dinamicamente (sem `-static`), o resultado é 13% mais lento no Linux. Note que você ainda pode utilizar uma biblioteca do MySQL dinamicamente ligada à sua aplicação cliente. É só o servidor que é crítico para performance.
- Se você corta seu binário `mysqld` com `strip libexec/mysqld`, o binário gerado pode ficar até 4% mais rápido.
- Para uma conexão de um cliente para um servidor em execução na mesma máquina, se você conecta utilizando TCP/IP em vez de utilizar um arquivo socket Unix, o rendimento é 7.5% mais lento no mesmo computador. (Se você fizer conexão à `localhost`, o MySQL irá, por padrão, utilizar sockets).
- Para conexões TCP/IP de um cliente para um servidor, conectando a um servidor remoto em outra máquina será 8-11% mais lento que conectando ao servidor local na mesma máquina, mesmo para conexões Ethernet de 100M.
- Quando executar o nosso teste de benchamrk usando conexões seguras (todos os dados crptografados com suporte interno SSL) ele se torna 55% mais lento.
- Se você compilar com `--with-debug=full`, a maioria das consultas será 20% mais lentas. Algumas consultas podem demorar muito mais tempo (por exemplo, os benchmarks do MySQL demonstram 35% de perda). Se utilizar `--with-debug`, a queda será de apenas 15%. Para uma versão do `mysqld` compilada com `--with-debug=full`, você pode desabilitar a verificação de memória em tempo de execução iniciando-o com a opção `--skip-safemalloc`. O resultado final neste caso deve estar próximo de quando compilado com `--with-debug`.
- Em um Sun UltraSPARC-IIe, Forte 5.0 é 4% mais rápido que `gcc` 3.2.
- Em um Sun UltraSPARC-IIe, Forte 5.0 é 4% mais rápido em modo de 32 bits que em modo de 64 bits.
- Compilando com `gcc` 2.95.2 para o ultrasparc com a opção `-mcpu=v8 -Wa,-xarch=v8plusa` melhora a performance em 4%.
- No Solaris 2.5.1, a MIT-pthreads é 8-12% mais lenta do que as threads nativas do Solaris em um único processador. Com mais carga/CPUs a diferença deve aumentar.
- Executar com `--log-bin` deixa o `mysqld` 1 % mais lento.
- Compilando no Linux-x86 com gcc sem frame pointers `-fomit-frame-pointer` ou `-fomit-frame-pointer -ffixed-ebp` deixa o `mysqld` 1-4% mais rápido.

A distribuição MySQL-Linux fornecida pela MySQL AB é normalmente compilada com `pgcc`, mas vamos retornar ao uso do `gcc` pelo fato de um bug no `pgcc` que gera o código que não executa no AMD. Continuaremos a usar o `gcc` até que o bug seja resolvido. Neste meio tempo, se você possui uma máquina que não seja AMD, você pode ter um binário mais rápido compilando com o `pgcc`. O binário padrão do MySQL para Linux é ligado estaticamente para conseguir mais desempenho e ser mais portável.

5.5.4 Como o MySQL Utiliza a Memória

A lista abaixo indica algumas das maneiras inas quais o servidor `mysqld` utiliza a memória. Onde aplicável, o nome da variável do servidor relevante ao uso de memória é fornecido:

- O buffer de chave (variável `key_buffer_size`) é compartilhado por todas as threads; Outros buffers usados pelo servido são alocados quando necessários. Veja Seção 5.5.2 [Parâmetros de servidor], Página 454.
- Cada conexão utiliza algum espaço específico da thread: Uma de pilha (padrão de 64K, variável `thread_stack`), um buffer de conexão (variável `net_buffer_lenght`), e um buffer de resultados (variável `net_buffer_lenght`). Os buffers de conexões e resultados são aumentados dinamicamente para `max_allowed_packet` quando necessário. Quando uma consulta está sendo executada, uma cópia da string da consulta atual é também alocada.
- Todas as threads compartilhas a mesma memória base.
- Somente as tabelas `ISAM` e `MyISAM` compactadas são mapeadas em memória. Isto é porque o espaço de memória de 32-bits de 4GB não é grande o bastante para a maioria das grandes tabelas. Quando sistemas com endereçamento de 64-bits se tornarem comuns poderemos adicionar um suporte gieral para o mapeamento de memória.
- Cada requisição fazendo uma varredura sequencial em uma tabela aloca um buffer de leitura (variável `read_buffer_size`).
- Ao ler registros na ordem "randômica" (por exemplo, depois de uma ordenação) um buffer de leitura randômico é alocado para evitar pesquisas em disco. (variável `read_rnd_buffer_size`).
- Todas as joins são feitas em um único passo, e a maioria delas podem ser feitas mesmo sem usar uma tabela temporária. A maioria das tabelas temporárias são tabelas baseadas em memória (HEAP). Tabelas temporárias com uma grande extensão de registros (calculada como a soma do tamanho de todas as colunas) ou que contenham colunas `BLOB` são armazenadas em disco.

  Um problema nas versões do MySQL anteriores a 3.23.2 é que se uma tabela HEAP excede o tamanho de `tmp_table_size`, você recebe o erro `The table nome_tabela is full`. A partir da versão 3.23.2, isto é tratado alterando automaticamente a tabela em memória `HEAP` para uma tabela baseada em disco `MyISAM` quando necessário. Para contornar este problema, você pode aumentar o tamanho da tabela temporária configurando a opção `tmp_table_size` do `mysqld`, ou configurando a opção do SQL `SQL_BIG_TABLES` no progrma cliente. Veja Seção 5.5.6 [`SET OPTION`], Página 460. Na versão 3.20 do MySQL, o número máximo da tabela temporária é `record_buffer*16`; se você estiver utilizando esta versão, você terá que aumentar o valor `record_buffer`. Você também pode iniciar o `mysqld` com a opção `--big-tables` para sempre armazenar as tabelas temporárias em disco. Entretanto isto afetará a velocidade de várias consultas complicadas.
- A maioria das requisições que realizam ordenação alocam um bufer de ordenação e 0-2 arquivos temporários dependendo do tamanho do resultado. Veja Seção A.4.4 [Arquivos temporário], Página 929.
- Quase todas as análises e cálculos são feitos em um armazenamento de memória local. Nenhuma sobrecarga de memória é necessário para ítens pequenos e a alocação e liberação normal de memória lenta é evitada. A memória é alocada somente para grandes strings inesperadas; isto é feito com `malloc()` e `free()`.
- Cada arquivo de índice é aberto uma vez e o arquivo de dados é aberto uma vez para cada thread concorrente. Uma estrutura de tabela, estrutura de coluna para

cada coluna e um buffer de tamanho `3 * n` é alocado para cada thread concorrente. (onde `n` é o maior tamanho do registro, sem levar em consideração colunas `BLOB`. Uma coluna `BLOB` utiliza de 5 a 8 bytes mais o tamanho dos dados contidos na mesma. O manipulador de tabelas `ISAM`/`MyISAM` irão usar um registro extra no buffer para uso interno.

- Para cada tabela com colunas `BLOB`, um buffer é aumentado dinamicamente para ler grandes valores `BLOB`. Se você ler uma tabela, um buffer do tamanho do maior registro `BLOB` é alocado.
- Estruturas de manipulacão para todas tabelas em uso são salvos em um cache e gerenciado como FIFO. Normalmente o cache possui 64 entradas. Se uma tabela foi usada por duas threads ao mesmo tempo, o cache terá duas entredas para a tabela. Veja Seção 5.4.7 [Cache de tabela], Página 451.
- Um comando `mysqladmin flush-tables` fecha (ou instruções `FLUSH TABLES`) todas tabelas que não estão em uso e marca todas tabelas em uso para serem fechadas quando a thread atualmente em execução terminar. Isto irá liberar efetivamente a maioria da memória em uso.

`ps` e outros programas de informações do sistema podem relatar que o `mysqld` usa muita memória. Isto pode ser causado pelas pilhas de threads em diferentes endereços de memória. Por exemplo, a versão do `ps` do Solaris conta a memória não usada entre as pilhas como memória usada. Você pode verificar isto conferindo a memória disponível com `swap -s`. Temos testado o `mysqld` com detectores comerciais de perda de memória, portanto tais perdas não devem existir.

5.5.5 Como o MySQL Utiliza o DNS

Quando um novo cliente conecta ao `mysqld`, o `mysqld` extende uma nova thread para lidar com o pedido. Esta thread primeiro confere se o nome da máquina está no cache de nomes de máquinas. Se não, a thread tenta resolver o nome da máquina.

- Se o sistema operacional suporta as chamadas seguras com thread `gethostbyaddr_r()` e `gethostbyname_r()`, a thread as utiliza para fazer a resolução do nome máquina.
- Se o sistema operacional não suporta as chamadas de threads seguras, a thread trava um mutex e chama `gethostbyaddr()` e `gethostbyname()`. Perceba que neste caso nenhuma outra thread pode resolver outros nomes de máquinas que não existam no cache de nomes de máquina até que a primeira thread esteja destrave o mutex.

Você pode desabilitar a procura de nomes de máquinas no DNS iniciando o `mysqld` com a opção `--skip-name-resolve`. No entanto, neste caso você só pode usar números IP nas tabelas de privilégio do MySQL.

Se você possuir um DNS muito lento e várias máquinas, pode obter mais desempenho desligando a procura de nomes de máquinas usando a opção `--skip-name-resolve` ou aumentando `HOST_CACHE_SIZE` (valor padrão: 128) e recompilar `mysqld`.

Você pode desabilitar o cache de nomes de máquinas iniciando o servidor com a opção `--skip-host-cache`. Para limpar a cache do nome de máquinas, envie uma instru;ção `FLUSH HOSTS` ou execute o comando `mysqladmin flush-hosts`.

Se você deseja disabilitar as conexões `TCP/IP` totalmente, inicie o `mysqld` com a opção `--skip-networking`.

5.5.6 Sintaxe de SET

```
SET [GLOBAL | SESSION] sql_variable=expression,
    [[GLOBAL | SESSION] sql_variable=expression] ...
```

SET configura várias opções que afetam a operação do servidor ou seu cliente.

Os seguintes exemplos mostram as diferentes sintaxes que se pode usar para configurar variáveis:

Em versões antigas do MySQL permitiamos o uso da sintaxe SET OPTION, mas esta sintaxe agora está obsoleta.

No MySQL 4.0.3 adicionamos as opções GLOBAL e SESSION e acessamos as variáveis de inicialização mais importantes.

LOCAL pode ser usado como sinôniumo de SESSION.

Se você define diversas variáveis na mesma linha de comando, o último modo GLOBAL | SESSION é utilizado

```
SET sort_buffer_size=10000;
SET @@local.sort_buffer_size=10000;
SET GLOBAL sort_buffer_size=1000000, SESSION sort_buffer_size=1000000;
SET @@sort_buffer_size=1000000;
SET @@global.sort_buffer_size=1000000, @@local.sort_buffer_size=1000000;
```

A sintaxe @@nome_variável é suoprtada para tornar a sintaxe do MySQL compatível com outros bancos de dados.

As diferentes variáveis de sistema que podem ser configuradas estão descritas na seção de variáveis de sistema deste manual. Veja Seção 6.1.5 [Variáveis de sistema], Página 475.

Se você estiver usando SESSION (o padrão) a opção que você definir terá efeito até que o sessão atual finalize ou até que vecê atribua um valor diferente a esta opção. Se você estiver usando GLOBAL, que exige o privilégio SUPER, a opção é lembrada e usada pelas novas conexões até que o servidor reinicie. Se você quiser tornar uma opção permanente, você deve definí-la em um arquivo de opção. Veja Seção 4.1.2 [Arquivos de opções], Página 216.

Para evitar o uso incorreto, o MySQL exibirá um erro se você usar SET GLOBAL com uma variável que só pode ser usada com SET SESSION ou se você não estiver usando SET GLOBAL com uma variável global.

Se você quiser definir uma variável SESSION com um valor GLOBAL ou um valor GLOBAL ao valor padrão do MySQL, você pode configurá-lo com DEFAULT.

```
SET max_join_size=DEFAULT;
```

Isto é idêntico a:

```
SET @@session.max_join_size=@@global.max_join_size;
```

Se você quiser restringir o valor máximo com o qual uma variável de servidor pode ser configurado com o comando SET, você pode especificaá-lo usando a opção de linha de comando --maximum-variable-name. Veja Seção 4.1.1 [Opções de lina de comando], Página 207.

Você pode obter uma lista da maioria das variáveis com SHOW VARIABLES. Veja Seção 4.6.8.4 [SHOW VARIABLE], Página 309. Você pode obter o valor de uma variável específica com a sintaxe @@[global.|local.]variable_name:

```
SHOW VARIABLES like "max_join_size";
SHOW GLOBAL VARIABLES like "max_join_size";
SELECT @@max_join_size, @@global.max_join_size;
```

Segue aqui a descrição das variáveis que usam uma sintaxe `SET` não padrão e algumas das outras variáveis. A definição das outras variáveis podem ser encontrados na seção variáveis de sistema, entre as opções de inicialização ou na descrição de `SHOW VARIABLES`. Veja Seção 6.1.5 [Variáveis de sistema], Página 475. Veja Seção 4.1.1 [Opções de linha de comando], Página 207. Veja Seção 4.6.8.4 [SHOW VARIABLES], Página 309.

`AUTOCOMMIT= 0 | 1`

Se configurado com `1` todas alterações em uma tabela será feita de uma vez. Para iniciar uma transação de vários comandos, deve ser usada a instrução `BEGIN`. Veja Seção 6.7.1 [COMMIT], Página 615. Se configurado com `0` deve ser usado `COMMIT/ROLLBACK` para aceitar/recusar aquela transação. Veja Seção 6.7.1 [COMMIT], Página 615. Note que quando você altera do modo não-`AUTOCOMMIT` para `AUTOCOMMIT`, o MySQL irá fazer um `COMMIT` automático em quaisquer transações abertas.

`BIG_TABLES = 0 | 1`

Se definido com `1`, todas as tabelas temporárias são armazenadas no disco em vez de o ser na meória. Isto será um pouco mais lento, mas você não terá o erro `The table tbl_name is full` para grandes operações `SELECT` que exigem uma tabela temporária maior. O valor padrão para uma nova conexão é `0` (isto é, usa tabelas temporárias em memória) Esta opção era chamada `SQL_BIG_TABLES`. No MySQL 4.0 você normalmente nunca deve precisar deste parâmetro já que o MySQL converterá automaticamente tabelas em memória para tabelas em disco se isto for necessário.

`CHARACTER SET nome_conjunto_caracteres | DEFAULT`

Mapeia todas as strings do e para o cliente com o mapa especificado. Atualmente a única opção para `character_set_name` é `cp1251_koi8`, mas você pode adicionar novos mapas editando o arquivo '`sql/convert.cc`' na distribuição fonte do MySQL. O mapeamento padrão pode ser restaurado utilizando o valor `DEFAULT` para `character_set_name`.

Perceba que a sintaxe para configurar a opção `CHARACTER SET` é diferente da sintaxe para configurar as outras opções.

`DATE_FORMAT = format_str`

Determina como o servidor converte valores DATE para strings. Esta variável está disponível como uma opção global, local ou de linha de comando. `format_str` pode ser especificado convenientemente usando a função `GET_FORMAT()`. Veja Veja Seção 6.3.4 [Date and time functions], Página 529.

`DATETIME_FORMAT = format_str`

Determina como o servidor converte valores DATETIME para string. Esta variável está disponível como uma opção global, local ou de linha de comando. `format_str` pode ser especificada convenientemente usando a função `GET_FORMAT()`. Veja Veja Seção 6.3.4 [Date and time functions], Página 529.

`INSERT_ID = #`

Configura o valor que será usado pelo comando `INSERT` ou `ALTER TABLE` seguinte ao inserir um valor `AUTO_INCREMENT`. Isto é usado principalmente com o log de atualizações.

`LAST_INSERT_ID = #`

Configura o valor a ser retornado de `LAST_INSERT_ID()`. Ele é armazenado no log de atualizações quando você utiliza `LAST_INSERT_ID()` em um comando que atualiza uma tabela.

`LOW_PRIORITY_UPDATES = 0 | 1`

Se configurado com `1`, todas instruções `INSERT`, `UPDATE`, `DELETE` e `LOCK TABLE WRITE` irão esperar até que não existam `SELECT` ou `LOCK TABLE READ` pendentes na tabela afetada. Esta opção era chamada `SQL_LOW_PRIORITY_UPDATES`.

`MAX_JOIN_SIZE = value | DEFAULT`

Não permite que `SELECT`s que provavelmente necessitem examinar mais que `valor` combinações de registros. Configurando este valor, você pode obter `SELECT`s onde chaves não são usadas corretamente e que provavelmente gastarão um bom tempo. Configurando-o para um valor diferente do `DEFAULT` irá definir o atributo `SQL_BIG_SELECTS` com o padrão. Se você configurar o atributo `SQL_BIG_SELECTS` novamente, a variável `SQL_MAX_JOIN_SIZE` será ignorada. Você pode configurar um valor padrão para esta variável iniciando o `mysqld` com `-O max_join_size=#`. Esta opção era chamada `SQL_MAX_JOIN_SIZE`

Note que se o resultado da consulta ja estiver na cache de consultas, o verificaição acima não será feita. O MySQL irá enviar o resultado ao cliente. Uma vez que o resultado da consulta já foi consultado e não será responsabilidade do servidor enviar o resultado ao cliente.

`PASSWORD = PASSWORD('alguma senha')`

Configura a senha para o usuário atual. Qualquer usuário que não seja anônimo pode alterar sua própria senha!

`PASSWORD FOR user = PASSWORD('alguma senha')`

Configura a senha para um usuário específico no servidor atual. Somente um usuário com acesso ao banco de dados `mysql` pode fazer isto. O usuário deve ser fornecido no formato `usuário@home_maquina`, onde `usuário` e `nome_máquina` são exatamente o que estão listados nas colunas `User` e `Host` da tabela `mysql.user`. Por exemplo, se você possui uma entrada com os campos `User` e `Host` com `'bob'` e `'%.loc.gov'`, você escreveria:

```
mysql> SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('newpass');
```

Que é equivalente a:

```
mysql> UPDATE mysql.user SET Password=PASSWORD('newpass')
    ->                   WHERE User='bob' AND Host='%.loc.gov';
mysql> FLUSH PRIVILEGES;
```

`QUERY_CACHE_TYPE = OFF | ON | DEMAND`
`QUERY_CACHE_TYPE = 0 | 1 | 2`

Define a configuração da cache de consultas para esta thread. Set query cache setting for this thread.

Opção	Descrição
0 ou OFF	Não armazena ou recupera resultados.
1 ou ON	Armazena todos os resultados, exceto consultas `SELECT SQL_NO_CACHE ...`.
2 ou DEMAND	Armazena apenas consultas `SELECT SQL_CACHE ...`.

`SQL_AUTO_IS_NULL = 0 | 1`

Se configurado com `1` (padrão) o último registro inserido em uma tabela com um regitro auto_incremnto pode ser encontrado com a seguinte construção: `WHERE auto_increment_column IS NULL`. Isto é usado por alguns programas ODBC como o Access.

`SQL_BIG_SELECTS = 0 | 1`

Se configurado com `0`, o MySQL aborta as instruções `SELECT`s que provavelmente levam muito tempo (isto é, instruções para as quais o otimizador estima que o número de registros examinados provavelmente irá exceder o valor de `MAX_JOIN_SIZE`. Isto é útil quando uma instrução `WHERE` não aconselhada for utilizado. O valor padrão para uma nova conexão é `1` (que permitirá qualquer instrução `SELECT`).

Se você definir `MAX_JOIN_SIZE` com um valor diferente de `DEFAULT`, `SQL_BIG_SELECTS` será definida com `0`.

`SQL_BUFFER_RESULT = 0 | 1`

`SQL_BUFFER_RESULT` força para que o resultado das `SELECT`'s seja colocado em tabelas temporárias. Isto irá ajudar o MySQL a liberar mais cedos bloqueios de tabela e ajudarão em casos onde elas ocupam muito tempo para enviar o conjunto de resultados para o cliente.

`SQL_SAFE_UPDATES = 0 | 1`

Se configurado com `1`, o MySQL irá aborar se tentarmos fazer um `UPDATE` ou `DELETE` sem utilizar uma chave ou `LIMIT` na cláusula `WHERE`. Desta forma é possível capturar atualizações erradas ao criarmos comandos SQL manualmente.

`SQL_SELECT_LIMIT = valor | DEFAULT`

O número máximo de registros para retornar de instruções `SELECT`. Se uma `SELECT` tem uma cláusula `LIMIT`, o `LIMIT` tem precedêencia sobre o valor de `SQL_SELECT_LIMIT`. O valor padrão para uma nova conexão é "unlimited" (ilimitado). Se você alterou o limite, o valor padrão pode ser restaurado atribuindo o valor `DEFAULT` a `SQL_SELECT_LIMIT`.

`SQL_LOG_OFF = 0 | 1`

Se configurado com `1`, nenhum registro será feito no log padrão para este cliente, se o cliente tiver o privilégio `SUPER`.

`SQL_LOG_BIN = 0 | 1`

Se configurada com `0`, nenhum registro é feito no log binário para o cliente, se o cliente tiver o privilégio `SUPER`.

`SQL_LOG_UPDATE = 0 | 1`
: Se configurado com `0`, nenhum registro será feito no log de atualizações para o cliente, se o cliente tiver o privilégio `SUPPER`. Esta variável está obsoleta a partir da versão 5.0.

`SQL_QUOTE_SHOW_CREATE = 0 | 1`
: Se configurado com `1`, `SHOW CREATE TABLE` irá colocar os nomes de tabela e colunas entre aspas. Está **ligado** por padrão, para que replicação de tabelas com nomes de colunas estranhos funcione. Seção 4.6.8.8 [`SHOW CREATE TABLE`], Página 322.

`TIMESTAMP = valor_timestamp | DEFAULT`
: Configura a hora/data para este cliente. É usado para obter a hora e data original se você utiliza o log de atualizações para restaurar registros. `valor_timestamp` deve ser um timestamp UNIX Epoch, não um timestamp MySQL.

`TIME_FORMAT = format_str`
: Determina como o servidor converte valores `TIME` para string. Esta variável está disponível como uma opção global, local ou de linha de comando. `format_str` pode ser especificada convenientemente usando a função `GET_FORMAT()`. Veja Veja Seção 6.3.4 [Date and time functions], Página 529.

5.6 Detalhes de Disco

- Como mencionado acima, pesquisas em disco são o maior gargalo de desempenho. Estes problemas ficam cada vez mais aparentes quando os dados começam a crescer tanto que efetivo armazenamento em cache se torna impossível. Para grandes bancos de dados, onde você acessa dados mais ou menos aleatoriamente, você pode ter certeza de que precisará de pelo menos uma busca em disco para ler e várias para gravar os dados. Para minimizar este problema, utilize discos com menor tempo de pesquisa.
- Aumente o número de eixo de discos disponíveis (e então reduza a sobrecarga da pesquisa) ligando arquivos simbolicamente em diferentes discos ou utilizando striping de discos.

  Usando links simbólicos
  : Significa que, para tabelas MyISAM, você liga simbolicamente o índice e/ou arquivos de dados ao local comum no diretório de dados em outro disco (que pode também ser striped). Isto torna os tempos de pesquisa e leitura melhor (Se os discos não são usados para outras coisas). Veja Seção 5.6.1 [Links simbólicos], Página 465.

  Striping
  : Striping significa que você possui vários discos e coloca o primeiro bloco no primeiro disco, o segundo bloco no segundo disco, e o N-simo no (N módulo número_de_discos) disco, e assim por diante. Isto significa que se o seu tamanho de dados normais é menos que o tamanho do bloco (ou perfeitamente alinhado) você irá obter um desempenho muito melhor. Striping é muito dependente do SO e do tamanho do bloco. Portanto meça a performance de sua aplicação com diferentes tamanhos de blocos. Veja Seção 5.1.5 [Benchamrks], Página 423.

Perceba que a diferença de velocidade para striping é **muito** dependente dos parâmetros. Dependendo de como você configura os parâmetros do striping e do número de discos você pode obter uma diferença de várias ordens de grandeza. Note que você deve escolher a otimização randômica ou pelo acesso sequencial.

- Para confiabilidade você pode desejar utilizar RAID 0+1 (striping + espelhamento) mas neste caso você irá precisar de 2*N discos para armazenar N discos de dados. Isto é provavelmente a melhor opção se você possuir dinheiro! Você pode também, entretanto, ter que investir em algum software gerenciador de volumes para lidar com isto eficientemente.

  Uma boa opção é variar os níveis de RAID de acordo com a importância do dado. a Por exemplo, ter dados com alguma importância que podem ser regenerados em um armazenamento RAID 0 enquanto os dados realemtente importantes como informações de máquinas e logs em um sistema RAID 0+1 ou RAID de N discos. RAID N pode ser um problema se você tem várias escritas devido ao tempo para atualizar os bits de paridade.

- No Linux, você pode obter um desempenho muito melhor (cerca de 100% sobre carga pode ser comum) utilizando hdparm para configurar sua interface de disco! O exemplo a seguir deve ser muito útil para o MySQL (e provavelmente várias outras aplicações):

  ```
  hdparm -m 16 -d 1
  ```

  Perceba que o desempenho e confiança ao utilizar o exemplo acima depende de seu hardware, portanto nós sugerimos que você teste bem seu sistema depois de utilizar `hdparm`! Por favor consulte a página do manual (man) do `hdparm` para maiores informações! Se o `hdparm` não for usado corretamente, poderá resultar em corrupção do sistema de arquivos, assim realize backups de tudo antes de experimentar!

- Você pode também configurar os parâmetros para o sistema de arquivos que o banco de dados usa:

  -

    Se você não precisa saber quando os arquivos foram acessados pela última vez (o que é realmente útil em um servidor de banco de dados), você pode montar o seu sistema de arquivos com a opção `-o noatime`. Isto faz com que ele evite a atualização do último tempo de acesso no inode e com isto também evita algumas buscas em disco.

  - Em vários sistemas operacionais os discos podem ser montados com a opção 'async' para configurar o sistema de arquivos a ser atualizado de modo assíncrono. Se o seu computador é razoavelmente estável, isto deve fornecer mais desempenho sem sacrificar a segurança. (Esta opção é ligada por padrão no Linux.)

5.6.1 Utilizando Links Simbólicos

Você pode mover tabelas e bancos de dados do diretório de banco de dados para outras localizações e trocá-los por links simbólicas para os novos locais. Você pode fazer isto, por exemplo, para mover um banco de dados para um sistema de arquivos com mais espaço livre ou aumentar a velocidade de seu sistema esipalhando suas tabelas para discos diferentes.

A maneira recomendada de se fazer isto é ligar simbolicamente bancos de dados a discos diferentes e só ligar tabelas como último recurso.

5.6.1.1 Utilizando Links Simbólicos para Bancos de Dados

No Unix, a maneira de ligar simbolicamente um banco de dados é, primeiramente, criar um diretório em algum disco onde você possui espaço livre e então criar uma ligação simbólica para ele a partir do diretório do banco de dados do MySQL.

```
shell> mkdir /dr1/databases/test
shell> ln -s /dr1/databases/test mysqld-datadir
```

O MySQL não suporta que você ligue um diretório a vários bancos de dados. Trocando um diretório de banco de dados com uma ligação simbólica irá funcionar bem desde que não sejam feitos links simbólicos entre os bancos de dados. Suponha que você tenha um banco de dados `db1` sob o diretório de dados do MySQL, e então criar uma ligação simbólica `db2` que aponte para `db1`.

```
shell> cd /caminho/para/diretorio/dados
shell> ln -s db1 db2
```

Agora, para qualquer tabela `tbl_a` em `db1`, também aparecerá uma tabela `tbl_a` em `db2`. Se uma thread atualizar `db1.tbl_a` e outra atualizar `db2.tbl_a`, ocorrerão porblemas.

Se você realmente precisar disto, você deve alterar o código seguinte em '`mysys/mf_format.c`':

```
if (flag & 32 || (!lstat(to,&stat_buff) && S_ISLNK(stat_buff.st_mode)))
```

para

```
if (1)
```

No Windows você pode utilizar links simbólicos para diretórios compilando o MySQL com `-DUSE_SYMDIR`. Isto lhe permite colocar diferentes bancos de dados em discos diferentes. Veja Seção 5.6.1.3 [Links simbólicos no Windows], Página 468.

5.6.1.2 Utilizando Links Simbólicos para Tabelas

Antes do MySQL 4.0 você não deve utilizar tabelas com ligações simbólicas, se você não tiver muito cuidado com as mesmas. O problema é que se você executar `ALTER TABLE`, `REPAIR TABLE` ou `OPTIMIZE TABLE` em uma tabela ligada simbolicamente, os links simbólicos serão removidas e substituidos pelos arquivos originiais. Isto acontece porque o comando acima funcinoa criando um arquivo temporário no diretório de banco de dados e quando o comando é completo, substitui o arquivo original pelo arquivo temporário.

Você não deve ligar simbolicamente tabelas em um sistema que não possui uma chamada `realpath()` completa. (Pelo menos Linux e Solaris suportam `realpath()`

No MySQL 4.0 links simbólicos só são suportados completamente por tabelas `MyISAM`. Para outros tipos de tabelas você provavelmente obterá problemas estranhos ao fazer qualquer um dos comandos mencionados acima.

O tratamento de links simbólicos no MySQL 4.0 funciona da seguinte maneira (isto é mais relevante somente para tabelas `MyISAM`.

- No diretório de dados você sempre terá o arquivo de definições das tabelas e os arquivos de índice e o arquivo de dados. O arquivo de dados e o arquivo de índice podem ser movidos para qualquer lugar e substituidos no diretorio de dados pelos links simbólicos. O arquivo de definição não pode.
- Você pode ligar simbolicamente o arquivo índice e o arquivo de dados para diretórios diferentes, independente do outro arquivo.
- A ligação pode ser feita partir do sistema operacional (se o `mysqld` não estiver em execução) ou usando as opções `DATA DIRECTORY` ou `INDEX DIRECTORY` em `CREATE TABLE`. Veja Seção 6.5.3 [`CREATE TABLE`], Página 598.
- `myisamchk` não irá substituir um link simbólico pelo índice/arquivo. Ele funciona diretamente nos arquivos apontados pelos links simbólicos. Qualquer arquivo temporário será criado no mesmo diretório que o arquivo de dados/índice está.
- Quando você remove uma tabela que está usando links simbólicos, o link e o arquivo para o qual ela aponta são apagados. Esta é uma boa razão pela qual você **não** deve executar `mysqld` como `root` e não deve permitir que pessoas tenham acesso de escrita ao diretórios de bancos de dados do MySQL.
- Se você renomear uma tabela com `ALTER TABLE RENAME` e não deseja alterar o banco de dados, o link simbólico para o diretório de banco de dados será renomeada corretamente.
- Se você utiliza `ALTER TABLE RENAME` para mover uma tabela para outro banco de dados, então a tabela será movida para outro diretório de banco de dados e os links simbólicos antigos e os arquivos para os quais eles apontam serão removidos.
- Se você não utiliza links simbólicos, você deve usar a opção `--skip-symlink` do `mysqld` para garantir que ninguém pode usar `mysqld` para apagar ou renomear um arquivo fora do diretório de dados.

O que ainda não é suportado:

- `ALTER TABLE` ignora todas as opções de tabela `DATA DIRECTORY` e `INDEX DIRECTORY`.
- `SHOW CREATE TABLE` não relata se a tabela possui links simbólicos antes do MySQL 4.0.15. Isto também é verdade para `mysqldump` que usa `SHOW CREATE TABLE` para gerar instruções `CREATE TABLE`.
- `BACKUP TABLE` e `RESTORE TABLE` não respeitam links simbólicos.
- O arquivo `frm` **nunca** deve ser um link simbólico (como dito anteriormente, apenas os dados e índices podem ser links simbólicos). Fazer isto (por exemplo para fazer sinônimos), produzirá resultados errados. Suponha que você tenha um banco de dados `db1` sob o diretório de dados do MySQL, uma tabela `tbl1` neste banco de dados e você faça um link simbólico `tbl2` no diretório `db1` que aponmta para `tbl1`:

  ```
  shell> cd /path/to/datadir/db1
  shell> ln -s tbl1.frm tbl2.frm
  shell> ln -s tbl1.MYD tbl2.MYD
  shell> ln -s tbl1.MYI tbl2.MYI
  ```

  Agora se uma thread lê `db1.tbl1` e outra thread atualiza `db1.tbl2`, haverá problemas: a cache de consultas será enganada (ela acreditará que `tbl1` não foi atualizado e retornará resultados desatualizados), o comando `ALTER` em `tbl2` também irá falhar.

5.6.1.3 Usando Links Simbólicos para Bancos de Dados no Windows

A partir do MySQL versão 3.23.16, o `mysqld-max` e servidores `mysql-max-nt` na distribuição MySQL são compilados com a opção `-DUSE_SYMDIR`. Isto permite que você coloque um diretório de banco de dados em discos diferentes adicionando um link simbólico para ele. (Isto é parecido com o a com que links simbólicos funcionam no Unix, embora o procedimento para configurar o link seja diferente).

No Windows, você cria um link simbólico para um banco de dados MySQL criando um arquivo que contem o caminho para o diretório de destino. Salve o arquivo no diretório de dados usando o nome de arquivo '`nome_bd.sym`', onde `nome_bd` é o nome do banco de dados.

Por exemplo, se o diretório de dados do MySQL é '`C:\mysql\data`' e você precisa ter o banco de dados `foo` localizado em '`D:\data\foo`', você deve criar o arquivo '`C:\mysql\data\foo.sym`' que contêm o caminho `D:\data\foo\`. Depois disto, todas tabelas criadas no banco de dados `foo` serão criadas no '`D:\data\foo`'. O diretório '`D:\data\foo`' deve existir para ele funcionar. Note também que o link simbólico não será usado se um diretório com o nome do banco de dados existe no diretório de dados MySQL. Isto significa que se você já tem um diretório de banco de dados chamado '`foo`' no direorio de dados, você deve movê-lo para '`D:\data`' antes do link simbólico ser efetivado. (Para evitar problemas, o servidor não deve estar executando quando você mover o diretório do banco de dados.)

Note que devido a penalidade que você tem na velocidade quando abre todas as tabelas, nós não habilitamos esta opção por padrão, mesmo se você compilar o MySQL com suporte a isto. Para habilitar links simbólicos você deve colocar no seu arquivo `my.cnf` ou `my.ini` a seguinte entrada:

```
[mysqld]
symbolic-links
```

No MySQL 4.0 `--simbolic-links` está habilitado por padrão. Se você não precisa usá-lo você pode usar a opção `skip-symbolic-linkd`.

6 Referência de Linguagem do MySQL

O MySQL possui uma interface SQL muito complexa mas intuitiva e fácil de aprender. Este capítulo descreve os vários comandos, tipos e funções que você precisa conhecer para usar o MySQL de maneira eficiente e efetiva. Este capítulo também serve como referência para todas as funcionalidades incluídas no MySQL. Para poder utilizar este capítulo eficientemente, você deve achar útil fazer referência aos vários índices.

6.1 Estrutura da Linguagem

6.1.1 Literais: Como Gravar Strings e Numerais

Esta seção descreve as diversas maneiras para gravar strings e números no MySQL. Ela também cobre as várias nuances e "pegadinhas" pelas quais você pode passar ao lidar com estes tipos básicos no MySQL.

6.1.1.1 Strings

Uma string é uma sequência de caracteres, cercada por caracteres de aspas simples (‘’’) ou duplas (‘"’) (Se você utiliza o modo ANSI deve utilizar somente as aspas simples). Exemplos:

```
'uma string'
"outra string"
```

Em uma string, certas sequências tem um significado especial. Cada uma destas sequências começam com uma barra invertida (‘\’), conhecida como *caracter de escape*. O MySQL reconhece a seguinte sequência de escape:

`\0`	Um caracter ASCII 0 (`NUL`).
`\'`	Um caracter de aspas simples (‘’’).
`\"`	Um caracter de aspas duplas (‘"’).
`\b`	Um caracter de backspace.
`\n`	Um caracter de nova linha.
`\r`	Um caracter de retorno de carro.
`\t`	Um caracter de tabulação.
`\z`	ASCII(26) (Control-Z). Este caracter pode ser codificado para permitir que você contorne o problema que o ASCII(26) possui comoEND-OF-FILE ou EOF (Fim do arquivo) no Windows. (ASCII(26) irá causar problemas se você tentar usar `mysql banco_dados < nome_arquivo`).
`\\`	O caracter de barra invertida (‘\’) character.
`\%`	Um caracter ‘%’. Ele pode ser usado para pesquisar por instâncias literais de ‘%’ em contextos onde ‘%’ deve, de outra maneira, ser interpretado como um meta caracter. Veja Seção 6.3.2.1 [Funções de comparações de string], Página 519.

`\_` Um caracter '_'. Ele é usado para pesquisar por instâncias literais de '_' em contextos onde '_' deve, de outra maneira, ser intrerpretado como um meta caracter. Veja Seção 6.3.2.1 [Funções de comparações de string], Página 519.

Note que se você utilizar '`\%`' ou '`\_`' em alguns contextos de strings, eles retornarão as strings '`\%`' e '`\_`' e não '`%`' e '_'.

Estas são as várias maneiras de incluir aspas com uma string:

- Um '`'`' dentro de uma string com '`'`' pode ser escrita como '`''`'.
- Um '`"`' dentro de uma string com '`"`' pode ser escrita como '`""`'.
- Você pode preceder o caracter de aspas com um caracter de escape ('`\`').
- Um '`'`' dentro de uma string com '`"`' não precisa de tratamento especial e não precisa ser duplicada ou utilizada com caracter de escape. Da mesma maneira, '`"`' dentro de uma string com '`'`' não necessita de tratamento especial.

As instruções `SELECT` exibidas abaixo demonstram como citações e escapes funcionam:

```
mysql> SELECT 'hello', '"hello"', '""hello""', 'hel''lo', '\'hello';
+-------+---------+-----------+--------+--------+
| hello | "hello" | ""hello"" | hel'lo | 'hello |
+-------+---------+-----------+--------+--------+

mysql> SELECT "hello", "'hello'", "''hello''", "hel""lo", "\"hello";
+-------+---------+-----------+--------+--------+
| hello | 'hello' | ''hello'' | hel"lo | "hello |
+-------+---------+-----------+--------+--------+

mysql> SELECT "This\nIs\nFour\nlines";
+--------------------+
| This
Is
Four
lines |
+--------------------+
```

Se você deseja inserir dados binários em uma coluna `BLOB`, os caracteres a seguir devem ser representados por sequências de espace:

`NUL` ASCII 0. Você deve representá-lo como '`\0`' (uma barra invertida e um caractere '`0`').

`\` ASCII 92, barra invertida. Representado como '`\\`'.

`'` ASCII 39, aspas simples. Representado como '`\'`'.

`"` ASCII 34, aspas duplas. Representado como '`\"`'.

Se você escreve código C, você pode utilizar a função da API C `mysql_escape_string()` para caracteres de escape para a instrução `INSERT`. Veja Seção 12.1.2 [Visão geral da função API C], Página 778. No Perl, pode ser utilizado o método `quote` do pacote `DBI` para converter caracteres especiais para as sequências de escape corretas. Veja Seção 12.5.2 [Classe `DBI` do Perl], Página 881.

Deve ser utilizada uma função de escape em qualquer string que contêm qualquer um dos caracteres especiais listados acima!

Alternativamente, muitas APIs do MySQL fornecem algumas da capacidades de placeholder que permitem que você insira marcadores especiais em um string de consulta e então ligar os valores dos dados a eles quando você executa a consulta. Neste caso, a API inclui, automaticamente, os caracteres especiais de escape nos valores para você.

6.1.1.2 Números

Inteiros são representados como uma sequência de dígitos. Números de ponto flutuante utilizam '`.`' como um separador decimal. Ambos os tipos devem ser precedidos por '`-`' para indicar um valor negativo.

Exemplos de inteiros válidos:

```
1221
0
-32
```

Exemplo de números de ponto flutuante válidos:

```
294.42
-32032.6809e+10
148.00
```

Um inteiro pode ser usado em um contexto de ponto flutuante; ele é interpretado como o de ponto flutuante equivalente.

A partir da versão 4.1.0, a constante `TRUE` é avaliada com `1` e `FALSE` é avaliada com `0`.

6.1.1.3 Valores Hexadecimais

O MySQL suporta valores hexadecimais. No contexto numérico estes atuam como um inteiro (precisão de 64-bits). No contexto de strings, atuam como uma string binária onde cada par de dígitos hexadecimais é convertido para um caracter:

```
mysql> SELECT x'4D7953514C';
         -> MySQL
mysql> SELECT 0xa+0;
         -> 10
mysql> SELECT 0x5061756c;
         -> Paul
```

No MySQL 4.1 (e no MySQL 4.0 quando usado com a opçõa `--new`) o tipo padrão de um valor hexadecimal é uma string. Se você deseja estar certo que a string é tratado como um número, você pode usar `CAST( ... AS UNSIGNED)` no valor hexadecimal.

A sintaxe `x'stringhexa'` (nova na versão 4.0) é baseada no padrão SQL e a sintaxe `0x` é baseada no ODBC. Strings hexadecimeis são frequentemente usadas pelo ODBC para suprir valores para colunas `BLOB`. Você pode converter uma string ou um número no formato hexadecimal com a função `HEX()`.

6.1.1.4 Valores NULL

O valor NULL significa "sem dados" e é diferente de valores como 0 para tipos numéricos ou strings vazias para tipos string. Veja Seção A.5.3 [Problemas com NULL], Página 932.

NULL pode ser representado por \N ao usar o formato de arquivo texto para importação ou exportação (LOAD DATA INFILE, SELECT ... INTO OUTFILE). Veja Seção 6.4.8 [LOAD DATA], Página 588.

6.1.2 Nomes de Banco de dados, Tabela, Índice, Coluna e Alias

Nomes de banco de dados, tabela, índice, coluna e apelidos seguem todos as mesmas regras no MySQL.

Note que as regras foram alteradas a partir do MySQL versão 3.23.6, quando introduzimos aspas em identificadores (nomes banco de dados, tabela e coluna) com ‘‘’. ‘"’ funcionará também para citar identificadores se você executar no modo ANSI. Veja Seção 1.8.2 [Modo ANSI], Página 42.

Identificador	Tamanho máximo (bytes)	Caracteres permitidos
Banco de dados	64	Qualquer caractere que é permitido em um nome de diretório exceto ‘/’ ou ‘.’.
Tabela	64	Qualquer caractere permitido em um nome de arquivo, exceto ‘/’ ou ‘.’.
Coluna	64	Todos os caracteres.
Alias	255	Todos os caracteres.

Note que em adição ao mostrado acima, você não pode ter ASCII(0) ou ASCII(255) ou o caracter de citação (aspas) em um identificador.

Se o identificador é uma palavra restrita ou contêm caracteres especiais você deve sempre colocá-lo entre ‘ ao usá-lo:

```
mysql> SELECT * FROM ‘select‘ WHERE ‘select‘.id > 100;
```

Veja Seção 6.1.7 [Reserved words], Página 479.

Se você estiver executando o MySQL no modo MAXDB ou ANSI_QUOTES, ele também pode citar identificadores com aspas duplas:

```
mysql> CREATE TABLE "test" (col INT);
ERROR 1064: You have an error in your SQL syntax. (...)
mysql> SET SQL_MODE="ANSI_QUOTES";
mysql> CREATE TABLE "test" (col INT);
Query OK, 0 rows affected (0.00 sec)
```

Veja Seção 4.1.1 [Command-line options], Página 207.

Em versões do MySQL anteriores a 3.23.6, as regras se nomes eram as seguintes:

- Um nome pode consistir de caracteres alfanuméricos do conjunto atual de caractres e também ‘_’ e ‘$’. O conjunto de caracteres padrão é o ISO-8859-1 Latin1; e pode ser alterado com a opção --default-character-set no mysqld. Veja Seção 4.7.1 [Conjunto de caracteres], Página 325.

- Um nome pode iniciar com qualquer caractere que é legal no nome. Em particular, pode iniciar com um número (isto difere de vários outros sistemas de bancos de dados!). Entretanto um nome não pode consistir *somente* de números.
- O caractere '`.`' não pode ser utilizado em nomes porque ele é usado para extender o formato pelo qual você pode fazer referências a colunas (veja abaixo).

É recomendado que você não utilize nomes como `1e`, porque uma expressão como `1e+1` é ambígua. Ela pode ser interpretada como a expressão `1e + 1` ou como o número `1e+1`.

No MySQL você pode se referir a uma coluna utilizando uma das formas seguintes:

Coluna de referência	Significado
`nome_campo`	Coluna `nome_campo` de qualquer tabela usada na consulta contendo uma coluna com aquele nome.
`nome_tabela.nome_campo`	Coluna `nome_campo` da tabela `nome_tabela` do banco de dados atual.
`nome_bd.nome_tabela.nome_campo`	Coluna `nome_campo` da tabela `nome_tabela` do banco de dados `nome_bd`. Esta forma é disponível no MySQL Versão 3.22 ou posterior.
`‘nome_coluna‘`	Uma coluna que é uma palavra chave ou contem caracteres especiais.

Você não precisa especificar um prefixo de `nome_tabela` ou `nome_bd.nome_tabela` para uma referência de coluna em uma instrução, a menos que a referência seja ambígua. Por exemplo, suponha que cada tabela `t1` e `t2` contenham uma coluna `c`, e você deve recuperar `c` em uma instrução `SELECT` que utiliza ambas tabelas `t1` e `t2`. Neste caso, `c` é ambíguo porque ele não é único entre as tabelas usadas na instrução, portanto deve ser indicado qual é a tabela que se deseja escrever, `t1.c` ou `t2.c`. De mesma forma, se você for recuperar de uma tabela `t` em um banco de dados `db1` e uma tabela `t` em um banco de dados `db2`, você deve se refererir às colunas nestas tabelas como `db1.t.nome_campo` e `db2.t.nome_campo`.

A sintaxe `.nome_tabela` indica a tabela `nome_tabela` no banco de dados atual. Esta sintaxe é aceitada para compatibilidade ODBC, porque alguns programas ODBC prefixam os nomes das tabelas com um caracter '`.`'.

6.1.3 Caso Sensitivo nos Nomes

No MySQL, bancos de dados e tabelas correspondem a diretórios e arquivos em seus diretórios. Consequentemente, o caso sensitivo no sistema operacional irá determinar o caso sensitivo nos nomes de bancos de dados e tabelas. Isto significa que nomes de bancos de dados e tabelas são caso sensitivo na maioria dos Unix e caso insensitivo no Windows. Uma exceção proeminente aqui é o Mac OS X, quando o o sistema de arquivos padrão HPS+ está sendo usado. No entanto o Mac OS X também suporta volumes UFS, esle são caso sensitivo no Mac OS X assim como são no Unix. Veja Seção 1.8.3 [Extensões ao ANSI], Página 43.

NOTA: Apesar de nomes de bancos e tabelas serem caso insensitivo no Windows, você não deve fazer referência a um certo banco de dados ou tabela utilizando casos diferentes na mesma consulta. A consulta a seguir não deve funcionar porque ela chama uma tabela como `minha_tabela` e outra como `MINHA_TABELA`.

```
mysql> SELECT * FROM minha_tabela WHERE MINHA_TABELA.col=1;
```

Nomes de colunas não são caso sensitivo em todas as circunstâncias.

Aliases nas tabelas são caso sensitivo. A consulta seguinte não deve funcionar porque ela faz referência ao alias como `a` e como `A`.

```
mysql> SELECT nome_campo FROM nome_tabela AS a
           WHERE a.nome_campo = 1 OR A.nome_campo = 2;
```

Se você tem um problema para lembrar o caso usado para os nomes de tabelas, adote uma convenção consistente, como sempre criar bancos de dados e tabelas utilizando nomes em minúsculas.

Uma maneira para evitar este problema é iniciar o `mysqld` com `-O lower_case_nome_tabelas=1`. Por padrão esta opção é 1 no Windows e 0 no Unix.

Se `lower_case_nome_tabelas` for 1, o MySQL irá converte todos os nomes de tabelas para minúsculo no armazenamento e pesquisa. (A partir da versão 4.0.2, esta opção também se aplica ao nome do banco de dados. A partir da 4.1.1 isto também se aplica a alias de tabelas). Perceba que se você alterar esta opção, será necessário converter primeiramente seus nomes de tabelas antigos para minúsculo antes de iniciar o `mysqld`.

Se você mover os arquivos `MyISAM` do Windows pare o Unix, você pode, em alguns casos, precisar usar a ferramenta '`mysql_fix_extensions`' para corrigir o caso ad extensão do arquivo em cada diretório de banco de dados específico ('`.frm`' em letra minúscula, '`.MYI`' e '`.MYD`' em letras maiúsculas). '`mysql_fix_extensions`' pode ser encontado no subdiretório '`scripts`'.

6.1.4 Variáveis de Usuário

O MySQL suporta variáveis específicas da conexão com a sintaxe `@nomevariável`. Um nome de variável pode consiste de caracteres alfanuméricos do conjunto de caracteres atual e também '`_`', '`$`' e '`.`'. O conjunto de caracteres padrão é ISO-8859-1 Latin1; ele pode ser alterado com a opção `--default-character-set` do `mysqld`. Veja Seção 4.7.1 [Conjunto de caracteres], Página 325. Os nomes das variáveis de usuários são caso insensitivo nas versão >= 5.0 e caso sensitivo nas versões < 5.0.

As variáveis não precisam ser inicializadas. Elas contém `NULL` por padrão e podem armazenar um valor inteiro, real ou uma string. Todas as variáveis de uma thread são automaticamente liberadas quando uma thread termina.

Você pode configurar uma variavel com a syntaxe `SET`.

```
SET @variável= { expressao inteira | expressao real | expressao string }
[,@variável= ...].
```

Você também pode atribuir um valor a uma variável em outras instruções diferentes de `SET`. No entanto, neste caso o operador de atribuição é `:=` em vez de `=`, porque `=` é reservado para comparações em instruções diferentes de `SET`:

```
mysql> SET @t1=0, @t2=0, @t3=0;
mysql> SELECT @t1:=(@t2:=1)+@t3:=4,@t1,@t2,@t3;
+----------------------+------+------+------+
| @t1:=(@t2:=1)+@t3:=4 | @t1  | @t2  | @t3  |
+----------------------+------+------+------+
|                    5 |    5 |    1 |    4 |
+----------------------+------+------+------+
```

Variáveis de usuários devem ser utilizadas em expressões onde são permitidas. Isto não inclui utiliza-las em contextos onde um número é explicitamente necessário, assim como na cláusula `LIMIT` de uma instrução `SELECT` ou a clausula `IGNORE number LINES` de uma instrução `LOAD DATA`.

NOTE: Em uma instrução `SELECT` , cada expressão só é avaliada quando enviada ao cliente. Isto significa que nas cláusula `HAVING`, `GROUP BY`, ou `ORDER BY`, você não pode fazer referência a uma expreesão que envolve variáveis que são configuradas na instrução `SELECT`. Por examplo, a seguinte instrução NÃO funcionará como o esperado:

```
SELECT (@aa:=id) AS a, (@aa+3) AS b FROM nome_tabela HAVING b=5;
```

A razão é que o `@aa` não irá conter o valor da linha atual, mas o valor da `id` da linha previamente aceita.

A regra geral é nunca atribuir **e** usar a mesma variável na mesma instrução.

Outra questão com configurar uma variável e usá-la na mesma instrução é que o tipo do resultado padrão de uma variável é baseada no tipo da variável no início da instrução. (Assume-se que uma variável não atribuída possui o valor `NULL` e é do tipo STRING). O seguitne exemplo ilustra isto:

```
mysql> SET @a="test";
mysql> SELECT @a,(@a:=20) FROM table_name;
```

Neste caso o MySQL relatará ao cliente que a coluna 1 é uma string e converte todos os acessos de `@a` a strings, mesmo que `@a` seja configurada com um número para a segunda linha. Depois que a instrução é executada `@a` será considerado como um número.

Se você tiver qualquer problema com isto, evite tanto configurar e usar a mesma variável na mesma instrução ou configurar a variável com 0, 0.0 ou `""` antes de usá-la.

6.1.5 Variáveis de Sistema

A partir do MySQL 4.0.3 fornecemos melhor acesso a diversas variáveis de sistema e conexão. Pode-se alterar a maioria dele ser ter de desligar o servidor.

Exite dois tipos de variáveis de sistema: Específica de threads (ou específica da conexão), variáveis que estão apenas na conexão atual e variáveis globais que são usadas para conigurqar eventos globais. Variáveis globais também são usadas para configurar os valores iniciais da variável específica da thread correspondente a nova conexão.

Quando o `mysqld` inicia, todas as variáveis globais são inicialisadas a partir dos argumentos de linha de comando e arquivos de opção. Você pode alterar o valor com o comando `SET GLOBAL` command. Quando uma nova thread é criada, a variável específica da thread é iniciada a partir das variáveis globais e não alteram mesmo se você executar um novo comando `SET GLOBAL`.

Para definir os valor de uma variável `GLOBAL`, você deve usar uma das seguintes sintaxes: (Aqui usamos `sort_buffer_size` como uma variável exemplo).

```
SET GLOBAL sort_buffer_size=valor;
SET @@global.sort_buffer_size=valor;
```

Para definir o valor de uma variável `SESSION`, você pode usar uma das seguintes sintaxes:

```
SET SESSION sort_buffer_size=valor;
SET @@session.sort_buffer_size=valor;
SET sort_buffer_size=valor;
```

Se você não especificar `GLOBAL` ou `SESSION` então será usado `SESSION`. Veja Seção 5.5.6 [SET OPTION], Página 460.

`LOCAL` é um sinônimo para `SESSION`.

Para recuperar o valor de uma variável `GLOBAL` você pode usar um dos seguintes comandos:

```
SELECT @@global.sort_buffer_size;
SHOW GLOBAL VARIABLES like 'sort_buffer_size';
```

Para retornar o valor de uma variável `SESSION` você pode usar um dos seguintes comandos:

```
SELECT @@session.sort_buffer_size;
SHOW SESSION VARIABLES like 'sort_buffer_size';
```

Quando você **retorna** o valor de uma cariável com a sintaxe `@@nome_variável` e você não especificar `GLOBAL` ou `SESSION` então o MySQL retornará o valor específico da thread (`SESSION`), se ele existir. Se não, o MySQL retornará o valor global.

A razão da exigência de `GLOBAL` apenas para definir a variável `GLOBAL`, mas não para recuperá-la e assegurar que não criemos problemas posteriormente ao introduzirmos um variável específica da thread com o mesmo nome ou remover uma variável específica da thread. Neste caso, você pode acidentalmente alterar o estado do servidor como um todo, e não apenas em sua conexão.

A seguir apresentamos uma lista completa de todas as variáveis que altera e recupera se você pode usar `GLOBAL` ou `SESSION` com elas.

Nome Variável	**Tipo Valor**	**Tipo**
autocommit	bool	SESSION
big_tables	bool	SESSION
binlog_cache_size	num	GLOBAL
bulk_insert_buffer_size	num	GLOBAL | SESSION
concurrent_insert	bool	GLOBAL
connect_timeout	num	GLOBAL
convert_character_set	string	SESSION
delay_key_write	OFF | ON | ALL	GLOBAL
delayed_insert_limit	num	GLOBAL
delayed_insert_timeout	num	GLOBAL
delayed_queue_size	num	GLOBAL
error_count	num	SESSION
flush	bool	GLOBAL
flush_time	num	GLOBAL
foreign_key_checks	bool	SESSION
identity	num	SESSION
insert_id	bool	SESSION
interactive_timeout	num	GLOBAL | SESSION
join_buffer_size	num	GLOBAL | SESSION
key_buffer_size	num	GLOBAL

last_insert_id	bool	SESSION
local_infile	bool	GLOBAL
log_warnings	bool	GLOBAL
long_query_time	num	GLOBAL | SESSION
low_priority_updates	bool	GLOBAL | SESSION
max_allowed_packet	num	GLOBAL | SESSION
max_binlog_cache_size	num	GLOBAL
max_binlog_size	num	GLOBAL
max_connect_errors	num	GLOBAL
max_connections	num	GLOBAL
max_error_count	num	GLOBAL | SESSION
max_delayed_threads	num	GLOBAL
max_heap_table_size	num	GLOBAL | SESSION
max_join_size	num	GLOBAL | SESSION
max_relay_log_size	num	GLOBAL
max_sort_length	num	GLOBAL | SESSION
max_tmp_tables	num	GLOBAL
max_user_connections	num	GLOBAL
max_write_lock_count	num	GLOBAL
myisam_max_extra_sort_file_size	num	GLOBAL | SESSION
myisam_repair_threads	num	GLOBAL | SESSION
myisam_max_sort_file_size	num	GLOBAL | SESSION
myisam_sort_buffer_size	num	GLOBAL | SESSION
net_buffer_length	num	GLOBAL | SESSION
net_read_timeout	num	GLOBAL | SESSION
net_retry_count	num	GLOBAL | SESSION
net_write_timeout	num	GLOBAL | SESSION
query_cache_limit	num	GLOBAL
query_cache_size	num	GLOBAL
query_cache_type	enum	GLOBAL
read_buffer_size	num	GLOBAL | SESSION
read_rnd_buffer_size	num	GLOBAL | SESSION
rpl_recovery_rank	num	GLOBAL
safe_show_database	bool	GLOBAL
server_id	num	GLOBAL

slave_compressed_protocol	bool	GLOBAL
slave_net_timeout	num	GLOBAL
slow_launch_time	num	GLOBAL
sort_buffer_size	num	GLOBAL | SESSION
sql_auto_is_null	bool	SESSION
sql_big_selects	bool	SESSION
sql_big_tables	bool	SESSION
sql_buffer_result	bool	SESSION
sql_log_binlog	bool	SESSION
sql_log_off	bool	SESSION
sql_log_update	bool	SESSION
sql_low_priority_updates	bool	GLOBAL | SESSION
sql_max_join_size	num	GLOBAL | SESSION
sql_quote_show_create	bool	SESSION
sql_safe_updates	bool	SESSION
sql_select_limit	bool	SESSION
sql_slave_skip_counter	num	GLOBAL
sql_warnings	bool	SESSION
table_cache	num	GLOBAL
table_type	enum	GLOBAL | SESSION
thread_cache_size	num	GLOBAL
timestamp	bool	SESSION
tmp_table_size	enum	GLOBAL | SESSION
tx_isolation	enum	GLOBAL | SESSION
wait_timeout	num	GLOBAL | SESSION
warning_count	num	SESSION
unique_checks	bool	SESSION

Variáveis marcadas com `num` podem ter um valor numérico. Variáveis marcadas com `bool` podem ser definidas com 0, 1, `ON` ou `OFF`. Variáveis do tipo `enum` devem, normalmente, ser atribuídas com um dos valores disponíveis para a variável, mas podem também ser definidas com o número correspondente ao valor enum. (O primeiro valor enum é 0).

Aqui está uma descrição de algumas das variáveis:

Variáveis	**Descrição**
identity	Alias para last_insert_id (compatibilidade com Sybase)
sql_low_priority_updates	Alias para low_priority_updates
sql_max_join_size	Alias para max_join_size
version	Alias para VERSION() (compatibilidade com Sybase (?))

Uma descrição da outra definição de tabela pode ser encontrada na seção de opções de inicialização, na descrição de `SHOW VARIABLES` e na seção `SET`. Veja Seção 4.1.1 [Opções de linha de comando], Página 207. Veja Seção 4.6.8.4 [SHOW VARIABLES], Página 309. Veja Seção 5.5.6 [SET OPTION], Página 460.

6.1.6 Sintaxe de Comentários

O servidor MySQL suporta os estilos de comentário `# no fim da linha`, `-- no fim da linha` e `/* na linha ou em multiplas linhas */`

```
mysql> select 1+1;     # Este comentário continua até o fim da linha
mysql> select 1+1;     -- Este comnetário continua até o fim da linha
mysql> select 1 /* Este é um comentário de linha */ + 1;
mysql> select 1+
/*
Este é um comentário
de múltiplas linhas
*/
1;
```

Note que o estilo de comentário `--` requer que pelo menos um espaço após o código `--`!

Embora o servidor entenda as sintaxes de comentários aqui descritas, existem algumas limitções no modo que o cliente `mysql` analisa o comentário `/* ... */`:

- Caracteres de aspas simples e aspas duplas são utilizados para indicar o início de uma string com aspas, mesmo dentro de um comentário. Se as aspas não coincidirem com uma segunda aspas dentro do comentário, o analisador não percebe que o comentário tem um fim. Se você estiver executando o `mysql` interativamente, você pode perceber a confusão ocorrida por causa da mudança do prompt de `mysql>` para `'>` ou `">`.
- Um ponto e vírgula é utilizado para indicar o fim de uma instrução SQL e qualquer coisa que venha após ele indica o início da próxima instrução.

Estas limitações se aplicam tanto a quando se executa `mysql` interativamente quanto quando se coloca oos comandos em um arquivo e pede para que `mysql` leia as entradas deste arquivo com o comando `mysql < some-file`.

MySQL suporta o estilo de comentário SQL-99 '`--`' apenas se o segundo traço for seguido de espaço Veja Seção 1.8.4.7 [ANSI diff comments], Página 51.

6.1.7 Tratamento de Palavras Reservadas no MySQL

Um problema comum ocorre quando tentamos criar tabelas com nome de campo que usam nomes de tipos de dados ou funções criadas no MySQL, com `TIMESTAMP` ou `GROUP`, Você poderá fazer isso (por exemplo, `ABS` é um nome de campo permitido). No entanto espaços não são permitidos entre o nome da função e o caracter '`(`', assim a função pode ser distinguida de uma referência a um nome de coluna.

Se você iniciar o servidor com a opção `--ansi` ou `--sql-mode=IGNORE_SPACE`, o servidor permite que a chamada da função tenha um espaço entre um nome de função e o caracter '`(`' seguinte. Isto faz com que o nome da funçao seja tratado como uma palavra reservada; como um resultadom nomes de coluna que são o mesmo que o nome de uma função devem ser colocada entre aspas como descrito em Seção 6.1.2 [Legal names], Página 472.

As seguintes palavras são explicitamente reservadas em MySQL. Muitas delas são proibidas pelo ANSI SQL92 como nomes de campos e/ou tabelas. (por examplo, `group`). Algumas poucas são reservadasporque o MySQL precisa delas e está usando (atualmente) um analisador `yacc`:

Word	Word	Word
ADD	ALL	ALTER
ANALYZE	AND	AS
ASC	ASENSITIVE	AUTO_INCREMENT
BDB	BEFORE	BERKELEYDB
BETWEEN	BIGINT	BINARY
BLOB	BOTH	BY
CALL	CASCADE	CASE
CHANGE	CHAR	CHARACTER
CHECK	COLLATE	COLUMN
COLUMNS	CONDITION	CONNECTION
CONSTRAINT	CONTINUE	CREATE
CROSS	CURRENT_DATE	CURRENT_TIME
CURRENT_TIMESTAMP	CURSOR	DATABASE
DATABASES	DAY_HOUR	DAY_MICROSECOND
DAY_MINUTE	DAY_SECOND	DEC
DECIMAL	DECLARE	DEFAULT
DELAYED	DELETE	DESC
DESCRIBE	DETERMINISTIC	DISTINCT
DISTINCTROW	DIV	DOUBLE
DROP	ELSE	ELSEIF
ENCLOSED	ESCAPED	EXISTS
EXIT	EXPLAIN	FALSE
FETCH	FIELDS	FLOAT
FOR	FORCE	FOREIGN
FOUND	FRAC_SECOND	FROM
FULLTEXT	GRANT	GROUP
HAVING	HIGH_PRIORITY	HOUR_MICROSECOND
HOUR_MINUTE	HOUR_SECOND	IF
IGNORE	IN	INDEX
INFILE	INNER	INNODB
INOUT	INSENSITIVE	INSERT
INT	INTEGER	INTERVAL
INTO	IO_THREAD	IS
ITERATE	JOIN	KEY
KEYS	KILL	LEADING
LEAVE	LEFT	LIKE
LIMIT	LINES	LOAD
LOCALTIME	LOCALTIMESTAMP	LOCK
LONG	LONGBLOB	LONGTEXT
LOOP	LOW_PRIORITY	MASTER_SERVER_ID
MATCH	MEDIUMBLOB	MEDIUMINT
MEDIUMTEXT	MIDDLEINT	MINUTE_MICROSECOND
MINUTE_SECOND	MOD	NATURAL
NOT	NO_WRITE_TO_BINLOG	NULL
NUMERIC	ON	OPTIMIZE
OPTION	OPTIONALLY	OR

ORDER	OUT	OUTER
OUTFILE	PRECISION	PRIMARY
PRIVILEGES	PROCEDURE	PURGE
READ	REAL	REFERENCES
REGEXP	RENAME	REPEAT
REPLACE	REQUIRE	RESTRICT
RETURN	REVOKE	RIGHT
RLIKE	SECOND_MICROSECOND	SELECT
SENSITIVE	SEPARATOR	SET
SHOW	SMALLINT	SOME
SONAME	SPATIAL	SPECIFIC
SQL	SQLEXCEPTION	SQLSTATE
SQLWARNING	SQL_BIG_RESULT	SQL_CALC_FOUND_ROWS
SQL_SMALL_RESULT	SQL_TSI_DAY	SQL_TSI_FRAC_SECOND
SQL_TSI_HOUR	SQL_TSI_MINUTE	SQL_TSI_MONTH
SQL_TSI_QUARTER	SQL_TSI_SECOND	SQL_TSI_WEEK
SQL_TSI_YEAR	SSL	STARTING
STRAIGHT_JOIN	STRIPED	TABLE
TABLES	TERMINATED	THEN
TIMESTAMPADD	TIMESTAMPDIFF	TINYBLOB
TINYINT	TINYTEXT	TO
TRAILING	TRUE	UNDO
UNION	UNIQUE	UNLOCK
UNSIGNED	UPDATE	USAGE
USE	USER_RESOURCES	USING
UTC_DATE	UTC_TIME	UTC_TIMESTAMP
VALUES	VARBINARY	VARCHAR
VARCHARACTER	VARYING	WHEN
WHERE	WHILE	WITH
WRITE	XOR	YEAR_MONTH
ZEROFILL		

Os simbolos seguintes (da tabela acima) não são permitidos pelo SQL-99 mas permitidos pelo MySQL como nome de campos/tabelas. Isto ocorre porque alguns destes nomes são muito naturais e vários pessoas já o utilizaram.

- ACTION
- BIT
- DATE
- ENUM
- NO
- TEXT
- TIME
- TIMESTAMP

6.2 Tipos de Campos

MySQL suporta um certo números de tipos de campos que podem ser agrupaos em três categorias: tipos numéricos, tipos de data e hora, e tipos string (caracteres). Esta seção primeiro lhe dá uma visão geral dos tipos disponíveis e resume as exigencias de armazenamento em cada tipo de coluna, também fornece uma descrição mais detalhada da propriedade dos tipos em cada categoria. A visão dada é propositalmente breve. As descrições mais detalhdas devem ser consultadas para informações adicionais sobre tipos de campo particulares como os formatos permitidos nos quais você pode especificar valores.

Os tipos de campos suportados pelo MySQL estão listados abaixo: As seguintes letras são usadas como código nas descrições:

`M` Indica o tamanho máximo do display. O tamanho máximo oficial do display é 255.

`D` Aplica aos tipos de ponto flutuante e indica o número de digitos após o ponto decimal. O maior valor possível é 30, mas não pode ser maior que `M`-2.

Colchetes ('`[`' and '`]`') indicam partes de tipos específicos que são opicionais

Note que se você especificar `ZEROFILL` para um campo MySQL automaticamente irá adicionar o atributo `UNSIGNED` ao campo.

Aviso: você deve estar ciente de que quando fizer uma subtração entre valores inteiros, onde um deles é do tipo `UNSIGNED`, o resultado será sem sinal! Veja Seção 6.3.5 [Funções de coerção], Página 544.

`TINYINT[(M)] [UNSIGNED] [ZEROFILL]`
: Um inteiro muito pequeno. A faixa deste inteiro com sinal é de `-128` até `127`. A faixa sem sinal é de `0` até `255`.

`BIT`
`BOOL`
`BOOLEAN` Estes são sinônimos para `TINYINT(1)`.

O sinônimo `BOOLEAN` foi adicionado na versão 4.1.0.

Um tipo boolean verdadeiro será introduzido de acordo com o SQL-99.

`SMALLINT[(M)] [UNSIGNED] [ZEROFILL]`
: Um inteiro pequeno. A faixa do inteiro com sinal é de `-32768` até `32767`. A faixa sem sinal é de `0` a `65535`.

`MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]`
: Um inteiro de tamanho médio. A faica com sinal é de `-8388608` a `8388607`. A faixa sem sinal é de `0` to `16777215`.

`INT[(M)] [UNSIGNED] [ZEROFILL]`
: Um inteiro de tamanho normal. A faixa com sinal é de `-2147483648` a `2147483647`. A faixa sem sinal é de `0` a `4294967295`.

`INTEGER[(M)] [UNSIGNED] [ZEROFILL]`
: Este é um sinônimo para `INT`.

`BIGINT[(M)] [UNSIGNED] [ZEROFILL]`

Um inteiro grande. A faixa com sinal é de `-9223372036854775808` a `9223372036854775807`. A faixa sem sinal é de `0` a `18446744073709551615`.

Existem algumas coisas sobre campos `BIGINT` sobre as quias você deve estar ciente:

- Todas as operações aritiméticas são feitas usando valores `BIGINT` ou `DOUBLE` com sinal, não devemos utilçizar inteiros sem sinal maiores que `9223372036854775807` (63 bits) exceto com funções ded bit! Se você fizer isto, alguns dos últimos digitos no resultado podem estar errados por causa de erros de arredondamento na conversão de `BIGINT` para `DOUBLE`.

  O MySQL 4.0 pode tratar `BIGINT` nos seguintes casos:

  - Usar inteiros para armazenar grandes valores sem sinais em uma coluna `BIGINT`.
  - Em `MIN(big_int_column)` e `MAX(big_int_column)`.
  - Quando usar operadores (`+`, `-`, `*`, etc.) onde ambos os operandos são inteiros.
- Você pode armazenar valores inteiro exatos em um campo `BIGINT` aramzenando-os como string, como ocorre nestes casos não haverá nenhuma representação intermediaria dupla.
- '`-`', '`+`', e '`*`' serão utilizados em cálculos aritiméticos `BIGINT` quando ambos os argumentos forem valores do tipo `INTEGER`! Isto significa que se você multilicar dois inteiros grandes (ou obter resultados de funções que retornam inteiros) você pode obter resultados inesperados quando o resultado for maior que `9223372036854775807`.

`FLOAT(precisão) [UNSIGNED] [ZEROFILL]`

Um número de ponto flutuante. Não pode ser sem sinal. `precisão` pode ser `<=24` para um número de ponto flutuante de precisão simples e entre 25 e 53 para um número de ponto flutuante de dupla-precisão. Estes tipos são como os tipos `FLOAT` e `DOUBLE` descritos logo abaixo. `FLOAT(X)` tem o mesma faixa que os tipos correspondentes `FLOAT` e `DOUBLE`, mas o tamanho do display e número de casas decimais é indefinido.

Na versão 3.23 do MySQL, este é um verdadeiro valor de ponto flutuante. Em versões anteriores , `FLOAT(precisão)` sempre tem 2 casas decimais.

Note que o uso de `FLOAT` pode trazer alguns problemas inesperados como nos cálculos já que em MySQL todos são feitos com dupla-precisão. Veja Seção A.5.6 [Registros sem correspondentes], Página 934.

Esta sintaxe é fornecida para comptibilidade com ODBC.

`FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]`

Um número de ponto flutuante pequeno (precisão simples). Os valores permitidos estão entre `-3.402823466E+38` e `-1.175494351E-38`, `0` e entre `1.175494351E-38` e `3.402823466E+38`. Se `UNSIGNED` for especificado, valores negativos não são permitidos O `M` é a largura do display e o `D` é o número de

casas decimais. `FLOAT` sem um argumento ou `FLOAT(X)` onde `X` <=24 tende a um número de ponto flutuante de precisão simples.

`DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]`
: Um número de ponto flutuante de tamanho normal (dupla-precisão). Valores permitidos estão entre `-1.7976931348623157E+308` e `-2.2250738585072014E-308`, `0` e entre `2.2250738585072014E-308` e `1.7976931348623157E+308`. Se `UNSIGNED` for especificado, valores negativos não são permitidos. O `M` é a largura do display e o `D` é número de casa decimais. `DOUBLE` sem argumento ou `FLOAT(X)` onde 25 <= X <= 53 são números de ponto flutuante de dupla-precisão.

`DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL]`
`REAL[(M,D)] [UNSIGNED] [ZEROFILL]`
: Estes são sinônimos para `DOUBLE`.

`DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]`
: Um número de ponto flutuante não empacotado. Se comporta como um campo `CHAR`: "não empacotado" significa que o número é armazenado como uma string, usando um caracter para cada digito do valor. O ponto decimal e, para números negativos, o sinal de menos ('-'), não são contados em M (mas é reservado espaço para isto). Se `D` for 0, os valores não terão ponto decimal ou parte fracionária. A faixa máxima do valor `DECIMAL` é a mesma do `DOUBLE`, mas a faixa atual para um campo `DECIMAL` dado pode ser limitado pela escolha de `M` e `D`. Se `UNSIGNED` é especificado, valores negativos não são permitidos.

  Se `D` não for definido será considerado como 0. Se `M` não for definido é considerado como 10.

  Note que antes da versão 3.23 do MySQL o argumento `M` deve incluir o espaço necessário para o sinal é o ponto decimal.

`DEC[(M[,D])] [UNSIGNED] [ZEROFILL]`
`NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]`
`FIXED[(M[,D])] [UNSIGNED] [ZEROFILL]`
: Este é um sinônimo para `DECIMAL`.

  O alias `FIXED` foi adicionado na versão 4.1.0 para compatibilidade com outros servidores.

`DATE`
: Uma data. A faixa suportada é entre `'1000-01-01'` e `'9999-12-31'`. MySQL mostra valores `DATE` no formato `'AAAA-MM-DD'`, mas permite a você a atribuir valores a campos `DATE` utilizando tanto strings quanto números. Veja Seção 6.2.2.2 [DATETIME], Página 491.

`DATETIME`
: Um combinação de hora e data. A faixa suportada é entre `'1000-01-01 00:00:00'` e `'9999-12-31 23:59:59'`. MySQL mostra valores `DATETIME` no formato `'AAAA-MM-DD HH:MM:SS'`, mas permite a você que atribuir valores a campos `DATETIME` utilizado strings ou números. Veja Seção 6.2.2.2 [DATETIME], Página 491.

`TIMESTAMP[(M)]`

Um timestamp. A faixa é entre `'1970-01-01 00:00:00'` e algum momento no ano `2037`.

No MySQL 4.0 ou anteriores, os valores `TIMESTAMP` são exibidos nos formatos `YYYYMMDDHHMMSS`, `YYMMDDHHMMSS`, `YYYYMMDD`, ou `YYMMDD`, dependendo se `M` é `14` (ou não definido), `12`, `8` ou `6`, mas permite a você atribuir valores ao campo `TIMESTAMP` usando strings ou números.

Um campo `TIMESTAMP` é util para gravar a data e a hora em uma operação de `INSERT` or `UPDATE` porque é automaticamente definido a data e a hora da operação mais recente se você próprio não especificar um valor. Você também pode definir a data e a hora atual atribuindo ao campo um valor `NULL`. Veja Seção 6.2.2 [Tipos de hora e data], Página 489.

Desde o MySQL 4.1, `TIMESTAMP` é retornado com um string com o formato `'YYYY-MM-DD HH:MM:SS'`. Se você deseja tê-lo como um número você deve adcionar `+0` a coluna timestamp. Teimestamp de tamanhos diferentes não são supoortados. Desde a versão 4.0.12, a opção `--new` pode ser usada para fazer o servidor se comportar como na versào 4.1.

Um `TIMESTAMP` sempre é armazenado em 4 bytes. O argumento `M` só afeta como a coluna `TIMESTAMP` é exibida.

Note que colunas do tipo `TIMESTAMP(M)` columns onde M é 8 ou 14 são apresentadas como números enquanto as outras colunas `TIMESTAMP(M)` são strings. Isto é apenas para assegurar que podemos eliminar e restaurar com segurança tabelas com estes tipos! Veja Seção 6.2.2.2 [DATETIME], Página 491.

`TIME`

Uma hora. A faixa é entre `'-838:59:59'` e `'838:59:59'`. MySQL mostra valores `TIME` no formato `'HH:MM:SS'`, mas permite a você atribuir valores para as colunas `TIME` usando strings ou números. Veja Seção 6.2.2.3 [TIME], Página 495.

`YEAR[(2|4)]`

Um ano no formato de 2 ou 4 digitos (padrão são 4 digitos). Os valores permitidos estão entre `1901` e `2155`, `0000` no formato de 4 digitos, e 1970-2069 se você estiver usando o formato de 2 digitos (70-69). MySQL mostra valores `YEAR` no formato `YYYY`, mas permie atribuir valores aos campos do tipo `YEAR` usando strings ou números. (O tipo `YEAR` é novo na versão 3.22 do MySL). Veja Seção 6.2.2.4 [YEAR], Página 496.

`[NATIONAL] CHAR(M) [BINARY | ASCII | UNICODE]`

Uma string de tamanho fixo que é sempre preenchida a direita com espaços até o tamanho especificado quando armazenado. A faixa de `M` é de 1 a 255 caracteres. Espaços extras são removidos quando o valor é recuperado. Valores `CHAR` são ordenados e comparados no modo caso insensitivo de acordo com o conjunto de caracteres padrão, a menos que a palavra chave `BINARY` seja utilizada.

A partir da versão 4.1.0, se o valor `M` especificado é maio que 255, o tipo de coluna é convertido para `TEXT`. Este é um recurso de compatibilidade.

`NATIONAL CHAR` (ou em sua forma reduzida `NCHAR`) é o modo SQL-99 de definir que um campo `CHAR` deve usar o conjunto CHARACTER padrão. Este é o padrão no MySQL.

`CHAR` é uma simplificação para `CHARACTER`.

A partir da versão 4.1.0, o atributo `ASCII` pode ser especificado o que atribui o conjunto de caracteres `latin1` a coluna `CHAR`.

A partir da versão 4.1.1, o atributo `UNICODE` pode ser especificado o que atribui o conjunto de caracteres `ucs2` a coluna `CHAR`.

O MySQL lhe permite criar um campo do tipo `CHAR(0)`.Isto é muito útil quando você precisa de comptibilidade com aplicativos antigos que dependem da existência de uma coluna, mas que, na verdade, não utiliza um valor. Isto também é muito bom quando você precisa de uma coluna que só pode receber 2 valores. Um `CHAR(0)`, que não é definido como um `NOT NULL`, só irá ocupar um bit e pode assumir 2 valores: `NULL` or `""`. Veja Seção 6.2.3.1 [CHAR], Página 497.

`BIT`
`BOOL`

`CHAR` This is a synonym for `CHAR(1)`.

`[NATIONAL] VARCHAR(M) [BINARY]`
Uma string de tamanho variável. **NOTA:** Espaços extras são removidos quando o caracter é armazenado (o que difere da especificação ANSI SQL). A faixa de `M` é de 1 a 255 characters. Valores `VARCHAR` são ordenados e comparados no modo caso insensitivo a menos que a palavra chave `BINARY` seja utilizada. Veja Seção 6.5.3.1 [Silent column changes], Página 607.

A partir da versão 4.1.0, se o valor `M` especificado é maio que 255, o tipo de coluna é convertido para `TEXT`. Este é um recurso de compatibilidade.

`VARCHAR` é uma simplificação para `CHARACTER VARYING`. Veja Seção 6.2.3.1 [CHAR], Página 497.

`TINYBLOB`
`TINYTEXT`

Um campo `BLOB` ou `TEXT` com tamanho máximo de 255 (2^8 - 1) caracteres. Veja Seção 6.5.3.1 [Silent column changes], Página 607. Veja Seção 6.2.3.2 [BLOB], Página 498.

`BLOB`
`TEXT`

Um campo `BLOB` ou `TEXT` com tamanho máximo de 65535 (2^16 - 1) caracteres. Veja Seção 6.5.3.1 [Silent column changes], Página 607. Veja Seção 6.2.3.2 [BLOB], Página 498.

`MEDIUMBLOB`
`MEDIUMTEXT`

Um campo `BLOB` ou `TEXT` com tamanho máximo de 16777215 (2^24 - 1) caracteres. Veja Seção 6.5.3.1 [Silent column changes], Página 607. Veja Seção 6.2.3.2 [BLOB], Página 498.

`LONGBLOB`
`LONGTEXT`

Um campo `BLOB` ou `TEXT` com tamanho máximo de 4294967295 ou 4G (2^32 - 1) caracteres. Veja Seção 6.5.3.1 [Silent column changes], Página 607. Veja Seção 6.2.3.2 [BLOB], Página 498. Até a versão 3.23 o protocolo cliente/servidor e tabelas MyISAM tinham um limite de 16M por pacote de transmissão/registro de tabela, a partir da versão 4.x o tamanho máximo permitido das colunas `LONGTEXT` ou `LONGBLOB` depende do tamanho máximo configurado para o pacote no protocolo cliente/servidor e da memória disponível. Veja Seção 6.2.3.2 [BLOB], Página 498.

`ENUM('valor1','valor2',...)`

Uma enumeração. Um objeto string que só pode ter um valor, selecionado da lista de valores `'valor1'`, `'valor2'`, `...`, `NULL` ou valor especial de erro `""`. Um `ENUM` pode ter um máximo de 65535 valores diferentes. Veja Seção 6.2.3.3 [ENUM], Página 499.

`SET('valor1','valor2',...)`

Um conjunto. Um objeto string que pode ter zero ou mais valores, cada um deve ser selecionado da lista de valores `'valor1'`, `'valor2'`, `....` Um `SET` pode ter até 64 membros. Veja Seção 6.2.3.4 [SET], Página 500.

6.2.1 Tipos Numéricos

MySQL suporta todos os tipos numéricos da ANSI/ISO SQL92. Estes tipos incluem o tipos de dados numéricos exatos (`NUMERIC`, `DECIMAL`, `INTEGER`, e `SMALLINT`), assim como o tipos de dados numéricos aproximados (`FLOAT`, `REAL`, e `DOUBLE PRECISION`). A palavra-chave `INT` é um sinônimo para `INTEGER`, e a palavra-chave `DEC` é um sinônimo para `DECIMAL`.

Os tipos `NUMERIC` e `DECIMAL` são implementados como o mesmo tipo pelo MySQL, como permitido pelo padrão SQL92. Eles são usados por valores para os quais é importante preservar a exatidão como, por exemplo, dados monetários. Quando é declarado um campo de algum desses tipos a precisão e a escala podem ser (e normalmente é) especificadas; por exemplo:

```
salario DECIMAL(5,2)
```

Neste exemplo, `5` (`precisão`) representa o número de digitos decimais significantes que serão armazenados no valor, e `2` (`escala`) representa o número de dígitos que serão armazenados após o ponto decimal. Neste caso, no entanto, a faixa de valores que podem ser armazendos na coluna `salario` é de `-99.99` a `99.99`. (MySQL pode, na verdade, armazenar numeros acima de `999.99` neste campo porque ele não precisa armazenar o sinal para números positivos).

Em ANSI/ISO SQL92, a sintaxe `DECIMAL(p)` é equivalente a `DECIMAL(p,0)`. Da mesma forma, a sintaxe `DECIMAL` é equivalente a `DECIMAL(p,0)`, onde a implementação permite decidir o valor de `p`. MySQL ainda não suporta nenhuma dessas duas formas variantes dos tipos de dados `DECIMAL/NUMERIC`. Este, geralmente, não é um problema sério, já que os principais benefícios destes tipos derivam da habilidade de controlar precisão e escala explicitamente.

Valores `DECIMAL` e `NUMERIC` são armazenados como strings, ao invés de um número de ponto-flutuante binário, para preservar o precisão decimal destes valores. Um caracter é usado para cada digito, para o ponto decimal (se `escala > 0`), e para o sinal '`-`' (para números negativos). Se `escala` é 0, valores `DECIMAL` e `NUMERIC` não contém ponto decimal ou parte fracionária.

A faixa máxima dos valores `DECIMAL` e `NUMERIC` é o mesmo do `DOUBLE`, mas a faixa real para um campo `DECIMAL` or `NUMERIC` pode ser limitado pela `precisão` ou pela `escala` para uma dada coluna. Quando é atribuído a uma coluna um valor com mais digitos após o ponto decimal do que o permitido especificado na `escala`, o valor é arredondado para aquela `escala`. Quando é atribuido um valor a uma coluna `DECIMAL` ou `NUMERIC` o qual excede a faixa determinada pelas `precisão` e `escala` especificada (ou padrão), MySQL armazena o valor correspondente ao final daquela faixa.

Como uma extensão do padrão ANSI/ISO SQL92, MySQL também suporta os tipos integrais `TINYINT`, `MEDIUMINT`, e `BIGINT` como listado nas tabelas abaixo. Outra extensão suportada pelo MySQL é especificar, opcionalmente, o tamanho do display de um valor inteiro entre parenteses seguindo o nome do tipo (por exemplo, `INT(4)`). Esta especificação opcional do tamanho é usada para preenchimento a esquerda do display de valores cujo tamanho é menor que o especificado para a coluna, mas não limita a faixa de valores que podem ser armazendos na coluna, nem o número de dígitos que serão mostrados para valores que excederem o tamanho especificado na coluna. Quando usados em conjunto com o atributo opcional de extensão `ZEROFILL`, o padrão do preenchimento de espaços é a substituição por zeros. Por exemplo, para uma coluna declarada com `INT(5) ZEROFILL`, o valor `4` é retornado como `00004`. Note que se você armazenar valores maiores que a largura do display em um coluna do tipo inteiro, você pode ter problemas quando o MySQL gerar tabelas temporárias para algum join complicado, já que nestes casos o MySQL acredita que os dados cabem na largura original da coluna.

Todos os tipos inteiros podem ter um atributo opcional (não-padrão) `UNSIGNED`. Valores sem sinal podem ser usados quando você permite apenas números positivos em uma coluna e você precisa de uma faixa de valores um pouco maior para a coluna.

Desde o MySQL 4.0.2, tipos de ponto flutuante também podem ser sem sinal (`UNSIGNED`). Como no tipos inteiros, este atributoprevine que valores negativos sejam armazenados na coluna. Ao contrário dos tipos negativos, o valor máximo da faixa permitida permanece o mesmo.

O tipo `FLOAT` é usado para representar tipos de dados numéricos aproximados. O padrão SQL-92 permite uma especificação opcional da precisão (mas não da faixa do expoente) em bits, após a a palavra `FLOAT` e entre parenteses. A implementação MySQL também suporta esta especificação opcional de precisão. Quando `FLOAT` é usada para uma tipo de coluna sem especificação de precisão, MySQL utiliza quatro bytes para armazenar os valores. Uma sintaxe variante também é suportada, com dois numeros entre parenteses após a palavra `FLOAT`. Com esta opção, o primeiro número continua a representar a quantidade de bytes necessária para armazenar o valor, e o segundo número especifica o número de dígitos a serem armazenados e mostrados após o ponto decimal (como com `DECIMAL` e `NUMERIC`). Quando é pedido ao MySQL para armazenar um número em uma coluna com mais digitos decimais após o ponto decimal que o especificado para esta coluna, o valor é arredondado eliminando os digitos extras quando armazenado.

Os tipos `REAL` e `DOUBLE PRECISION` não aceitam especificações de precisão. Como uma extensão do padrão SQL-92, o MySQL reconhece `DOUBLE` como um sinônimo para o tipo `DOUBLE PRECISION`. Em constraste com a exigencia do padrão de que a precisão do tipo `REAL` seja menor que aquele usado pelo `DOUBLE PRECISION`, MySQL implementa ambos como valores de ponto flutuante de 8 bits de dupla precisão (quando não estiver executando em "modo ANSI"). Para uma portabilidade máxima, códigos que requerem armazenamento de valores de dados numéricos aproximados usam `FLOAT` ou `DOUBLE PRECISION` sem especificação de precisão ou de numeros decimais.

Quando solicitado a armazenar um valor em uma coluna numérica que está fora da faixa permitida pelo tipo da coluna, o MySQL ajusta o valor ao limite da faixa permitida mais apropriado e armazena este valor.

Por exemplo, a faixa de uma coluna `INT` é de `-2147483648` a `2147483647`. Se você tentar inserir `-9999999999` em uma coluna `INT`, o valor é ajustado para o limite mais baixo da faixa de valores e `-2147483648` é armazenado. Da mesma forma, se você tentar inserir `9999999999`, `2147483647` será armazenado.

Se o campo `INT` é `UNSIGNED`, o tamanho da faixa do campo é o mesmo mas o limite passa a ser de `0` a `4294967295`. Se você tentar armazenar `-9999999999` e `9999999999`, os valores armazenados na coluna serão `0` e `4294967296`.

Conversões que ocorrem devido a ajustes são relatados como "avisos" para `ALTER TABLE`, `LOAD DATA INFILE`, `UPDATE`, e instruções `INSERT` multi-registros.

Tipo	**Bytes**	**De**	**Até**
`TINYINT`	1	-128	127
`SMALLINT`	2	-32768	32767
`MEDIUMINT`	3	-8388608	8388607
`INT`	4	-2147483648	2147483647
`BIGINT`	8	-9223372036854775808	9223372036854775807

6.2.2 Tipos de Data e Hora

Os tipos de data e hora são `DATETIME`, `DATE`, `TIMESTAMP`, `TIME`, e `YEAR`. Cada um desses tipos tem uma faixa de valores legais, assim com um valor "zero" que é usado quando você especifica um valor ilegal. Note que o MySQL permite que você armazene certos valores de datas inexistentes, como `1999-11-31`. A razão para isto é que pensamos que é responsabilidade do aplicativo tratar das verificações de data, não do servidor SQL. Para fazer uma verificação 'rápida' de data, MySQL só checa se o mês está na faixa de 0-12 e o dia está na faixa de 0-31. As faixas acima são definidas desta forma porque MySQL lhe permite armazenar, em um campo `DATE` ou `DATETIME`, datas onde o dia ou o dia/mês são zero. Isto é extremamente útil para aplicativos que precisam armazenar uma data de nascimento na qual você não sabe a data exata. Nestes casos você simplesmente armazena a data como `1999-00-00` ou `1999-01-00`. (Você não pode esperar obter um valor correto para funções como `DATE_SUB()` ou `DATE_ADD` para datas como estas.)

Aqui estão algumas considerações para ter em mente quando estiver trabalhando com tipos de data e hora.

- MySQL recupera valores para um tipo de data ou hora dado em um formato padrão, mas ele tenta interpretar uma variedade de formatos para os valores fornecidos (por

exemplo, quando você especifica um valor a ser atribuido ou comparado a um tipo de data ou hora). No entanto, só os formatos descritos na seção seguinte são suportados. É esperado que você forneça valores permitidos. Resultados imprevisiveis podem ocorrer se você usar outros formatos.

- Embora o MySQL tente interpretar valores em diversos formatos, ele sempre espera que a parte da data referente ao ano esteja mais a esquerda do valor. Datas devem ser dadas na ordem ano-mês-dia (por exemplo, `'98-09-04'`), ao invés das ordens mais usadas mês-dia-ano ou dia-mês-ano (por exemplo: `'09-04-98'`, `'04-09-98'`).
- MySQL converte automaticamente um tipo de data ou hora em um número se o valor é usado em um contexto numérico, e vice-versa.
- Quando o MySQL encontra um valor para um tipo de data ou hora que está fora da faixa permitida ou é ilegal neste tipo (veja o início desta seção), ele converte o valor para "zero". (A exceção ocorre no campo `TIME`, onde o valor fora da faixa é ajustado para o valor limite apropriado na faixa de valores deste tipo.) A tabela abaixo mostra o formato do valor "zero" para cada tipo:

Tipo de Coluna	**Valor "Zero"**
`DATETIME`	`'0000-00-00 00:00:00'`
`DATE`	`'0000-00-00'`
`TIMESTAMP`	`00000000000000` (tamanho depende do tamanho do display)
`TIME`	`'00:00:00'`
`YEAR`	`0000`

- Os valores "zero" são especiais, mas você pode armazenar ou fazer referência a eles explicitamente usando os valores mostrados na tabela. Você também pode fazer into usando `'0'` ou `0`, o que é mais fácil de escrever.
- Valores "zero" para data ou hora usados em **MyODBC** são convertidos automaticamente para `NULL` na versão 2.50.12 **MyODBC** e acima, porque ODBC não pode tratar tais valores.

6.2.2.1 Assuntos referentes ao ano 2000 (Y2K) e Tipos de Data

O MySQL tem sua própria segurança para o ano 2000 (veja Seção 1.2.5 [compatibilidade com o ano 2000], Página 11), mas os dados entrados no MySQL podem não ter. Qualquer entrada contendo valores de ano de 2 digitos é ambíguo, porque o século é desconhecido. Tais valores devem ser interpretados na forma de 4 digitos já que o MySQL armazena anos internamente utilizando 4 digitos.

Para tipos `DATETIME`, `DATE`, `TIMESTAMP` e `YEAR`, MySQL interpreta datas com valores ambíguos para o ano usando as seguintes regras:

- Valores de ano na faixa `00-69` são convertidos para `2000-2069`.
- Valores de anos na faixa `70-99` são convertidos para `1970-1999`.

Lembre-se de que essas regras fornecem apenas palpites razoáveis sobre o que a sua data significa. Se a heurística usada pelo MySQL não produz o valor você deve fornecer entre sem ambiguidade contendo valores de ano de 4 digitos.

`ORDER BY` irá ordenar tipos `YEAR/DATE/DATETIME` de 2 digitos apropriadamente.

Note também que algumas funções com `MIN()` e `MAX()` irão converter `TIMESTAMP/DATE` para um número. Isto significa que um timestamp com ano de 2 digitos não irá funcionar corretamente com estas funções. A solução neste caso é converter o `TIMESTAMP/DATE` para um formato de ano de 4 digitos ou usar algo como `MIN(DATE_ADD(timestamp,INTERVAL 0 DAYS))`.

6.2.2.2 Os Tipos `DATETIME`, `DATE` e `TIMESTAMP`

Os tipos `DATETIME`, `DATE`, e `TIMESTAMP` são relacionados. Esta seção descreve suas características, como eles se assemelham ou como se diferem.

O tipo `DATETIME` é usado quando você precisa de valores que contém informações sobre data e a a hora. MySQL recupera e mostra valores `DATETIME` no formato `'YYYY-MM-DD HH:MM:SS'`. A faixa suportada é de `'1000-01-01 00:00:00'` até `'9999-12-31 23:59:59'`. ("Suportada" significa que embora valores anteriores possam funcionar, não há nenhura garantia de disto.)

O tipo `DATA` é usado quando se necessita apenas do valor da data, sem a parte da hora. MySQL recupera e mostra valores do tipo `DATA` no formato `'YYYY-MM-DD'`. A faixa suportada é de `'1000-01-01'` até `'9999-12-31'`.

A coluna do tipo `TIMESTAMP` possui comportamento e propriedade variado, dependendo da versão do MySQL e do modo SQL que o servidor está executando.

Comportamento do `TIMESTAMP` ao executar no modo `MAXDB`

Quando o MySQL está executando no modo `SQPDB`, o `TIMESTAMP` comporta como `DATETIME`. Nenhuma atualização automática da coluna `TIMESTAMP` ocorre, como descrito no parágrafo seguinte. O MySQL pode ser executado no modo `MAXDB` a partir da versão 4.1.1. Veja Seção 4.1.1 [Command-line options], Página 207.

Comportamento do `TIMESTAMP` quando não está executando no modo `MAXDB`

O tipo de campo `TIMESTAMP` fornece um tipo que pode ser usado para, automaticamente, marcar operações `INSERT` or `UPDATE` com a data e hora atual. Se você tiver multiplas colunas `TIMESTAMP`, só a primeira é atualizada automaticamente.

Atualizações automaticas da primeira coluna `TIMESTAMP` ocorrem sob qualquer uma das seguintes condições:

- A coluna não é explicitamente especificada em uma instrução `INSERT` ou `LOAD DATA INFILE`.
- A coluna não é explicitamente especificada em uma instrução `UPDATE` e e alguma outra coluna muda o valor. (Note que um `UPDATE` que coloca em uma coluna o mesmo valor que ele já possui não irá causar a atualização da coluna `TIMESTAMP`, porque se você atribui a uma coluna o seu valor atual, MySQL ignora a atualização para maior eficiência).
- Você define explicitamente a uma coluna `TIMESTAMP` o valor `NULL`.

Outras colunas `TIMESTAMP`, além da primeira podem ser definidas com a data e hora atuais. Basta defini-las com `NULL` ou `NOW()`

Você pode definir colunas `TIMESTAMP` com um valor diferente da data e hora atuais colocando explicitamente o valor desejado. Isto é verdade mesmo para a primeira coluna `TIMESTAMP`. Você pode usar esta propriedade se, por exemplo, você quiser que um `TIMESTAMP` tenha seu valor definido como a data e hora atuais na criação de registros, mas não quer alterá-los quando o registro for atualizado mais tarde:

- Deixe o MySQL definir a coluna quando o registro é criado. Isto irá inicializa-la com a data e hora atuais.
- Quando você realizar subsequentes atualizações em outras colunas do registro, defina explicitamente a coluna `TIMESTAMP` com o valor atual.

Por outro lado, você pode achar que é mais fácil usar uma coluan `DATETIME` que você inicializa com `NOW()` quando o registro for criado e deixa como está em atualizações subsequentes.

Propriedades `TIMESTAMP` quando executando no modo `MAXDB`

Quando o MySQL está executando no modo `MAXDB`, `TIMESTAMP` é idêntico ao `DATETIME`. Ele usa o mesmo formato para armazenar e mostrar valores, e ele tem a mesma faixa. O MySQL pode ser executado no modo `MAXDB` a partir da versão 4.1.1. Veja Seção 4.1.1 [Command-line options], Página 207.

Propriedades `TIMESTAMP` a partir do MySQL 4.1 quando não executado no modo `MAXDB`

No MySQL 4.1.0, colunas `TIMESTAMP` são armazenadas e mostradas no mesmo formato que colunas `DATETIME`. Isto também significa que ele não podem ser estreitados ou alargados nos modos descritos no parágrafo seguinte. Em outras palavras, você não pode usar `TIMESTAMP(2)`, `TIMESTAMP(4)`, etc. Em outros casos, as propriedades são as mesmas de versões MySQL anteriores.

Propriedades `TIMESTAMP` antes do MySQL 4.1

Valores `TIMESTAMP` podem ter valores do incio de 1970 até algum momento do ano 2037, com a resolução de um segundo. Valores são mostrados como números

O formato no qual o MySQL recupera e mostra valores `TIMESTAMP` depende do tamanho do display, como ilustrado pela tabela que se segue: O formato 'cheio' `TIMESTAMP` é de 14 digitos, mas colunas `TIMESTAMP` podem ser criadas com tamanho de display menores:

Tipo da Coluna	**Formato do Display**
`TIMESTAMP(14)`	`YYYYMMDDHHMMSS`
`TIMESTAMP(12)`	`YYMMDDHHMMSS`
`TIMESTAMP(10)`	`YYMMDDHHMM`
`TIMESTAMP(8)`	`YYYYMMDD`
`TIMESTAMP(6)`	`YYMMDD`
`TIMESTAMP(4)`	`YYMM`
`TIMESTAMP(2)`	`YY`

Todas as colunas `TIMESTAMP` tem o mesmo tamanho de armazenamento, independente do tamanho de display. Os tamanhos de display mais comuns são 6, 8, 12, e 14. Você pode especificar um tamanho de display arbitrario na hora da criação da tabela, mas valores de 0 ou maiores que 14 são mudados para 14. Valores ímpares de tamanho na faixa de 1 a 13 são mudados para o maior número par mais próximo.

Nota: Na versão 4.1, `TIMESTAMP` é retornado com uma string com o formato `'YYYY-MM-DD HH:MM:SS'`, e timestamp de diferentes tamamnhos não são mais suportados.

Você pode especificar calores `DATETIME`, `DATE` e `TIMESTAMP` usando qualquer conjunto de formatos comum:

- Como uma string nos formatos `'YYYY-MM-DD HH:MM:SS'` ou `'YY-MM-DD HH:MM:SS'`. Uma sintaxe "relaxada" é permitida—nenhum caracter de pontuação pode ser usado como um delimitador entre parte de data ou hora. Por exemplo, `'98-12-31 11:30:45'`, `'98.12.31 11+30+45'`, `'98/12/31 11*30*45'`, e `'98@12@31 11^30^45'` são equivalentes.
- Como uma string nos formatos `'YYYY-MM-DD'` ou `'YY-MM-DD'`. Uma sintaxe "relaxada" é permitida aqui também. Por exemplo, `'98-12-31'`, `'98.12.31'`, `'98/12/31'`, e `'98@12@31'` são equivalentes.
- Como uma string sem delimitadores nos formatos `'YYYYMMDDHHMMSS'` ou `'YYMMDDHHMMSS'`, desde que a string faça sentido como data. Por example, `'19970523091528'` e `'970523091528'` são interpretadas com `'1997-05-23 09:15:28'`, mas `'971122129015'` é ilegal (tem uma parte de minutos sem sentido) e se torna `'0000-00-00 00:00:00'`.
- Como uma string sem delimitadores nos formatos `'YYYYMMDD'` ou `'YYMMDD'`, desde que a string tenha sentido com data. Por exemplo, `'19970523'` e `'970523'` são interpretedas como `'1997-05-23'`, mas `'971332'` é ilegal (tem uma parte de mês sem sentido) e se torna `'0000-00-00'`.
- Como um número nos formatos `YYYYMMDDHHMMSS` ou `YYMMDDHHMMSS`, desde que o número faça sentido como uma data. Por exemplo, `19830905132800` e `830905132800` são interpretedos como `'1983-09-05 13:28:00'`.
- Como um número nos formatos `YYYYMMDD` ou `YYMMDD`, desde que o número faça sentido como data. Por exemplo, `19830905` e `830905` são interpretedos como `'1983-09-05'`.
- Como o resultado de uma função que retorne uma valor aceitavel em um contexto `DATETIME`, `DATE` ou `TIMESTAMP`, tal como `NOW()` ou `CURRENT_DATE`.

Valores `DATETIME`, `DATE`, ou `TIMESTAMP` ilegais são convertidos para o valor "zero" do tipo apropriado (`'0000-00-00 00:00:00'`, `'0000-00-00'`, ou `00000000000000`).

Para valores especificados com strings que incluem delimitadores de data, não é necessário especificar dois digitos para valores de mês ou dia qua são menores que `10`. `'1979-6-9'` é o mesmo que `'1979-06-09'`. Similarmente, para valores especificados como strings que incluem delimitadores de hora, não é necessário especificar dois digitos para valores de hora, minutos ou segundo que são menores que `10`. `'1979-10-30 1:2:3'` Ré o mesmo que `'1979-10-30 01:02:03'`.

Valores especificados como números devem ter 6, 8, 12, ou 14 digitos. Se o número é de 8 ou 14 digitos, ele assume estar no formato `YYYYMMDD` ou `YYYYMMDDHHMMSS` e que o ano é dado pelos 4 primeiros dígitos. Se o é de 6 ou 12 dígitos, ele assume estar no formato

`YYMMDD` or `YYMMDDHHMMSS` e que o ano é dado pelos 2 primeiros digitos. Números que não possua estes tamanho são interpretados como calores preenchidos com zero até o tamanho mais próximo.

Valores especificados como strings não delimitadas são interpretados usando o seu tamanho como dado. Se a string possui 8 ou 14 caracteres, o ano é assumido como os 4 primeiros caracteres. De outra forma o assume-se que o ano são os 2 primeiros caracteres. A string é interpretadada esquerda para direita para encontrar os valores do ano, mês, dia, hora, minute e segundo, para as partes da string. Isto significa que você não deve utilizar strings com menos de 6 caracteres. Por exemplo, se você especificar `'9903'`, pensando em representar Março de 1999, você perceberá que o MySQL insere uma data "zero" em sua tabela. Isto ocorre porque os valores do ano e mês são `99` e `03`, mas a parte contendo o dia não existe (zero), então o valor não é uma data legal. No entanto, a partir do MySQL 3.23, você pode especificar explicitamente um valor de zero para representar dia ou mês faltantes. Por exemplo, você pode usar `'990300'` para inserir o valor `'1999-03-00'`.

Colunas `TIMESTAMP` armazena valores legais utilizando precisão total com a qual os valores foram especificados, independente do tamanho do display. Isto tem diversas implicações:

- Sempre especifique o ano, mês e dia, mesmo se seus tipos de coluna são `TIMESTAMP(4)` ou `TIMESTAMP(2)`. De outra forma, os valores não serão datas legais date e um `0` será armazenado.
- Se você usa `ALTER TABLE` para aumentar uma coluna `TIMESTAMP`, informações serão mostradas como se antes estivessem "escondidas".
- De forma similar, reduzindo o tamanho de uma coluna `TIMESTAMP` não causa perda de informação, exceto no sentido de que menos informação aparece quando os valores são mostrados.
- Embora os valores `TIMESTAMP` sejam armazenados com precisão total, a única função que opera diretamente com o valor armazenado é `UNIX_TIMESTAMP()`. OUtras funções operam com o formato do valor recuperado Isto significa que não se pode usar funções como `HOUR()` or `SECOND()` a menos que a parte relevante do valor `TIMESTAMP` esteja incluído no valor formatado. POr exemplo, a parte `HH` de uma coluna `TIMESTAMP` não é mostrada a menos que o tamanho do display seja de pelo menos 10, logo tentar usar `HOUR()` em um valor `TIMESTAMP` menor produz um resultado sem significado.

Você pode, algumas vezes, atribuir valores de um tipo de data para um objeto de um diferente tipo de data. No entanto pode haver algumas alterações de valores ou perda de informação

- Se você atribuir um valor de `DATE` value a um objeto `DATETIME` ou `TIMESTAMP`, a parte da hora do valor resultante é definido como `'00:00:00'`, porque o vlaor `DATE` não contém informações de hora.
- Se você atribuir um valor `DATETIME` ou `TIMESTAMP` para um objeto `DATE`, a parte da hora do valor resultante é deletado, pois o tipo `DATE` não armazena informações de hora.
- Lembre-se de que embora todos os valores `DATETIME`, `DATE`, e `TIMESTAMP` possam ser especificados usando o mesmo conjunto de formatos, os tipos não tem a mesa faixa de valores. Por exemplo, valores `TIMESTAMP` não podem ser anteriores a `1970` ou posteriores a `2037`. Isto significia que datas como `'1968-01-01'`, são permitidas como valores

`DATETIME` ou `DATE`, mas não são válidas para valores `TIMESTAMP` e serão covertidas para `0` se atribuidas para tais objetos.

Esteja ciente de certas dificuldades quando especificar valores de data:

- A forma "relaxada" permitida em valores especificados com strings podem causar certas confusões. Por exemplo, um valor como `'10:11:12'` pode parecer com um valor de hora devido ao limitador '`:`', mas se usado em um contexto de data será interpretado como o ano `'2010-11-12'`. O valor `'10:45:15'` será convertido para `'0000-00-00'` pois `'45'` não é um valor de mês permitido.
- O servidor MySQL funciona basicamente checando a validade da data: dias entre `00-31`, mês entre `00-12`, anos entre `1000-9999`. Qualquer data que não esteja nesta faixa será revetida para `0000-00-00`. Por favor, note que isto ainda lhe permite armazenar datas invalidas tais como `2002-04-31`. Isto permite a aplicações web armazenar dados de um formulário sem verificações adicionais. Para assegurar que a data é valida, faça a checagem em sua aplicação.
- Valores de anos especificados com 2 digitos são ambíguos, pois o século não é conhecido. MySQL interpreta valores de anos com dois digitos usando as seguintes regras:
  - Valores de ano na faixa de `00-69` são convertidos para `2000-2069`.
  - Valores de ano na faixa de `70-99` são convertidos para `1970-1999`.

6.2.2.3 O Tipo `TIME`

O MySQL recupera e mostra valores `TIME` no formato `'HH:MM:SS'` (ou no formato `'HHH:MM:SS'` para valores grandes). Volares `TIME` podem estar na faixa de `'-838:59:59'` até `'838:59:59'`. A razão para a parte da hora ser tão grande é que o tipo `TIME` pode ser usado não apenas para representar a hora do dia (que deve ser menor que 24 horas), mas também para tempo restante ou intervalos de tempo entre dois eventos(que podem ser maior que 24 horas ou mesmo negativo).

Você pode especificar valores `TIME` de variadas formas:

- Como uma string no formato `'D HH:MM:SS.fração'` . (Note que o MySQL não armazena ainda frações para a coluna time.) Pode-se também utilizar uma das seguintes sintaxes "relaxadas":

  `HH:MM:SS.fração`, `HH:MM:SS`, `HH:MM`, `D HH:MM:SS`, `D HH:MM`, `D HH` ou `SS`. Aqui `D` é um dia entre 0-33.
- Como uma string sem delimitadores no formato `'HHMMSS'`, desde que ela tenha sentido como uma hora. Por exemplo, `'101112'` é esntendido como `'10:11:12'`, mas `'109712'` é ilegal (a parte dos minutos não tem nenhum sentido) e se torna `'00:00:00'`.
- Como um número no formato `HHMMSS` , desde que tenha sentido como uma hora. Por exemplo, `101112` é entendido com `'10:11:12'`. Os formatos alternativos seguintes também são entendidos: `SS`, `MMSS`, `HHMMSS` e `HHMMSS.fração`. Note que o MySQL ainda não armazena frações.
- Como o resultado de uma função que retorne um valor que é aceitável em um contexto do tipo `TIME`, tal como `CURRENT_TIME`.

Para valores TIME especificados como uma string que incluem delimitadores de hora, não é necessário especificar dois dígitos para valores de hora, minutos ou segundos que sejam menores que 10. '8:3:2' é o mesmo que '08:03:02'.

Seja cuidadoso ao atribuir valores TIME "pequenos" para uma coluna TIME. Sem dois pontos, o MySQL interprete valores assumindo que os digitos mais a direita representam segundos. (MySQL interpreta valores TIME como tempo decorrido ao invés de hora do dia.) Por exemplo, você poderia pensar em '1112' e 1112 significam '11:12:00' (11 horas e 12 minutos), mas o MySQL o intepreta como '00:11:12' (onze minutos e 12 segundos). De forma similar, '12' e 12 são interpretados como '00:00:12'. Valores TIME com dois pontos, em contrapartida, são tratados como hora do dia. Isto é, '11:12' significará '11:12:00', não '00:11:12'.

Valores que são legais mas que estão fora da faixa permitidas são ajustados para o valor limita da faixa mais apropriado. Por exemplo, '-850:00:00' e '850:00:00' são convertidos para '-838:59:59' e '838:59:59', respectivmente.

Valores TIME ilegais são convertidos para '00:00:00'. Note que como '00:00:00' é um valor TIME, não temos com dizer, a partir de um valor '00:00:00' armazenado na tabela, se o valor original armazenado foi especificado como '00:00:00' ou se foi ilegal.

6.2.2.4 O Tipo YEAR

O tipo YEAR é um tipo de 1 byte usado para representar anos.

O MySQL recupera e mostra valores YEAR no formato YYYY. A faixa de valores é de 1901 até 2155.

Você pode especificar valores YEAR em uma variedade de formatos:

- Como uma string de 4 digitos na faixa de '1901' até '2155'.
- Como um número de 4 dígitos na faixa de 1901 até 2155.
- Como uma string de dis dígitos na faixa '00' até '99'. Valores na faixa de '00' até '69' e '70' até '99' são convetidas para valores YEAR na faixa de 2000 até 2069 e 1970 até 1999.
- Como um número de 2 digitos na faixa de 1 até 99. Valores na faixa de 1 até 69 e 70 até 99 são convertidos para valores YEAR na faixa de 2001 até 2069 e 1970 até 1999. Note que a faixa para números de dois dígitos é um pouco diferente da faixa de strings de dois dígitos, pois não se pode especificar zero diretamente como um número e tê-lo interpretado com 2000. Você **deve** especificá-lo como uma string '0' ou '00' ou ele será interpretado com 0000.
- Como o resultado de uma função que retorna um valor que é aceitável em um contexto do tipo YEAR, tal como NOW().

Valores YEAR ilegais são convertidos para 0000.

6.2.3 Tipos String

Os tipos strings são CHAR, VARCHAR, BLOB, TEXT, ENUM, e SET. Esta seção descreve como este tipos funcionam, suas exigências de armazenamento e como usá-los em suas consultas.

Tipo	Tam.maxímo	Bytes

`TINYTEXT` ou `TINYBLOB`	2^8-1	255
`TEXT` ou `BLOB`	2^16-1 (64K-1)	65535
`MEDIUMTEXT` ou `MEDIUMBLOB`	2^24-1 (16M-1)	16777215
`LONGBLOB`	2^32-1 (4G-1)	4294967295

6.2.3.1 Os Tipos `CHAR` e `VARCHAR`

Os tipos `CHAR` e `VARCHAR` são parecidos, mas diferem no modo como são armazenados e recuperados.

O tamanho de um campo `CHAR` é fixado pelo tamanho declarado na criação da tabela. O tamanho pode ser qualquer valor entre 1 e 255 (Como na versão 3.23 do MySQL, o tamanho pode ser de 0 a 255). Quando valores `CHAR` são armazenados, eles são preenchidos a direita com espaços até o tamanho especificado. Quando valores `CHAR` são recuperados, espaços extras são removidos.

Valores no campo `VARCHAR` são strings de tamanho variável. Você pode declarar um campo `VARCHAR` para ter qualquer tamanho entre 1 e 255, assim como para campo `CHAR`. No entanto, diferente de `CHAR`, valores `VARCHAR` são armazendos usando apenas quantos caracteres forem necessários, mais 1 byte para gravar o tamanho. Valores não são preenchidos; ao contrário, espaços extras são removidos quando valores são armazenados. (Esta remoção de espaços difere das especificações do SQL-99). Nenhum caso de conversão é feito durante um o armazenamento ou recuperação.

Se você atribuir um valor para uma coluna `CHAR` ou `VARCHAR` que exceda o tamanho máximo da coluna, o valor é truncado para este tamanho.

A seguinte tabela ilustra as diferenças entre os dois tipos de colunas, mostrando o resultado de se armazenar vários valores de strings em campos `CHAR(4)` e `VARCHAR(4)`:

Valor	`CHAR(4)`	**Exigência p/ armazenamento**	`VARCHAR(4)`	**Exigência p/ armazenamento**
`''`	`'  '`	4 bytes	`''`	1 byte
`'ab'`	`'ab '`	4 bytes	`'ab'`	3 bytes
`'abcd'`	`'abcd'`	4 bytes	`'abcd'`	5 bytes
`'abcdefgh'`	`'abcd'`	4 bytes	`'abcd'`	5 bytes

Os valores recuperados para as colunas `CHAR(4)` e `VARCHAR(4)` serão os mesmos em cada caso, já que espaços ectras são removidos das colunas `CHAR` quando recuperados.

Valores nas colunas `CHAR` e `VARCHAR` são ordenados e comparadaos no modo caso-insensitivo, a menos que o atributo `BINARY` seja especificado quando a tabela for criada. O atributo `BINARY` significa que os valores das colunas são ordenados e comparados no modo caso-sensitivo de acordo com a ordem ASCII da maquina onde o servidor MySQL está sesndo executado. `BINARY` não afeta como as colunas são armazenadas e recuperadas.

A partir da versão 4.1.0, o tipo de coluna `CHAR BYTE` é um alias para `CHAR BINARY`. Thite é um recurso para compatibilidade.

O atributo `BINARY` é pegajoso. Isto significa que se uma coluna definida com `BINARY` é usada na expressão, toda a expressão é comparada como um valor `BINARY`.

MySQL pode alterar sem aviso o tipo de uma coluna `CHAR` ou `VARCHAR` na hora de criar a tabela. Veja Seção 6.5.3.1 [Silent column changes], Página 607.

6.2.3.2 Os Tipos BLOB e TEXT

Um BLOB é um objeto binario grande que pode guardar um montante variado de dados. Os quatro tipos BLOB: TINYBLOB, BLOB, MEDIUMBLOB, e LONGBLOB diferem apenas no tamanho maximo dos valores que eles podem guradar. Veja Seção 6.2.6 [Storage requirements], Página 502.

Os quatro tipos TEXT: TINYTEXT, TEXT, MEDIUMTEXT, e LONGTEXT correspondem aos quatro tipos BLOB e têm o mesmo tamanho máximo e necessidade de tamanho para armazenamento. A única diferença entre os tipos BLOB e TEXT é que ordenação e comparação são realizadas no modo caso-sensitivo para valores BLOB e no modo caso-insensitivo para valores TEXT. Em outras palavras, um TEXT é um BLOB no modo caso-insensitivo. Nenhum caso de conversão é feito durante um o armazenamento ou recuperação.

Se você atribuir um valor a uma coluna BLOB ou TEXT que exceda o tamanho máximo do tipo da coluna, o valor é truncado para servir ao campo.

Em muitos casos, podemos considerar um campo TEXT como um campo VARCHAR que pode ser tão grande quando desejamos. Da mesma forma podemos considerar um campo BLOB como um campo VARCHAR BINARY. As diferenças são:

- Você pode ter indices em um campo BLOB e TEXT no MySQL Versão 3.23.2 e mais novas. Versões antigas do MySQL não suportam isto.
- Não há remoção de espaços extras para campos BLOB e TEXT quando os valores são armazenados, como há em campos VARCHAR.
- Colunas BLOB e TEXT não podem ter valores padrões.

A partir da versão 4.1.0, LONG e LONG VARCHAR mapeiam para o tipo de dados MEDIUMTEXT. Este é um recurso de compatibilidade.

MyODBC define valores BLOB como LONGVARBINARY e valores TEXT como LONGVARCHAR.

Como valores BLOB e TEXT podem ser extremamentes longos, você pode deparar com alguns problemas quando utilizá-los:

- Se você quiser utilizar GROUP BY ou ORDER BY em um campo BLOB ou TEXT, você deve converte-los em objetos de tamanho fixo. O modo padrão de se fazer isto é com a função SUBSTRING. Por exemplo:

```
mysql> SELECT comentario FROM nome_tabela,SUBSTRING(comentario,20) AS substr
    ->                  ORDER BY substr;
```

  Se você não fizer isto, só os primeiros max_sort_length bytes de uma coluna serão utilizados na ordenação. O valor padrão de max_sort_length é 1024; este calor pode ser alterado utilizando-se a opção -O quando o servidor é inicializado. Você pode agrupar uma expressão envolvendo valores BLOB ou TEXT especificando a posição da coluna ou utilizando apelidos (alias):

```
mysql> SELECT id,SUBSTRING(col_blob,1,100) FROM nome_tabela GROUP BY 2;
mysql> SELECT id,SUBSTRING(col_blob,1,100) AS b FROM nome_tabela GROUP BY b;
```

- O tamanho máximo de uma objeto BLOB ou TEXTé determinado pelo seu tipo, mas o maior valor que você pode, atualmente, transmitir entre o cliente e o servidor é determinado pela quantidade de memória disponível e o tamanho dos buffers de comunicação. Você pode mudar o tamanho do buffer de mensagem (max_allowed_packet), mas você deve faze-lo no servidor e no cliente. Veja Seção 5.5.2 [Server parameters], Página 454.

Note que cada valor `BLOB` ou `TEXT` é representado internamente por um objeto alocado searadamente. Está é uma diferença com todos os outros tipos de colunas, para o qual o armazenamento é alocado um por coluna quando a tabela é aberta.

6.2.3.3 O Tipo `ENUM`

Um `ENUM` é um objeto string cujo valor normalmente é escolhido de uma lista de valores permitidos que são enumerados explicitamente na especificação da coluna na criação da tabela.

O valor pode ser a string vazia (`""`) ou `NULL` sob certas circunstâncias:

- Se você inserir um valor inválido em um `ENUM` (isto é, uma string que não está presente na lista de valores permitidos), a string vazia é inserida no lugar como um valor especial de erro. Esta string pode se diferenciar de um string vazia 'norma' pelo fato de que esta string tem uo valor numérico 0. Veremos mais sobre este assunto mais tarde.
- Se um `ENUM` é declarado `NULL`, `NULL` é também um valor permitido para a coluna, e o valor padrao é `NULL`. Se um `ENUM` é decalarado `NOT NULL`, o valor padrão é o primeiro elemento da lista de valores permitidos.

Cada enumeração tem um índice:

- Valores da lista de elementos permitidos na especificação da coluna são números começados com 1.
- O valor de índice de uma string vazia que indique erro é 0. Isto significa que você pode usar a seguinte instrução `SELECT` para encontrar linhas nas quais valores `ENUM` inválidos forma atribuidos:

```
mysql> SELECT * FROM nome_tabela WHERE col_enum=0;
```

- O índice de um valor `NULL` é `NULL`.

Por exemplo, uma coluna especificada como `ENUM("um", "dois", "três")` pode ter quqlquer um dos valores mostrados aqui. O índice de cada valor também é mostrado:

Valor	**Indice**
`NULL`	`NULL`
`""`	0
`"um"`	1
`"dois"`	2
`"três"`	3

Uma enumeração pode ter um máximo de 65535 elementos.

A partir da versão 3.23.51 espaços extras são automaticamente deletados dos valores `ENUM` quando a tabela é criada.

O caso da letra é irrelevante quando você atribui valores a um coluna `ENUM`. No entanto, valores recuperados posteriormente da coluna terá o caso de letras de acordo com os valores que foram usados para especificar os valores permitidos na criação da tabela.

Se você recupera um `ENUM` em um contexto numérico, o indice do valor da coluna é retornado. Por exemplo, você pode recuperar valores numéricos de uma coluna `ENUM` desta forma:

```
mysql> SELECT col_enum+0 FROM nome_tabela;
```

Se você armazena um número em um `ENUM`, o número é tratado como um índice, e o valor armazenado é o membro da enumeração com este índice. (No entanto, este não irá funcionar com `LOAD DATA`, o qual trata todas as entradas como strings.) Não é aconselhável armazenar números em uma string `ENUM` pois pode tornar as coisas um pouco confusas.

Valores `ENUM` são armazenados de acordo com a ordem na qual os membros da enumeração foram listados na especificação da coluna. (Em outras palavras, valores `ENUM` são ordenados de acordo com o seus números de índice.) Por exemplo, `"a"` vem antes de `"b"` para `ENUM("a", "b")`, mas `"b"` vem antes de `"a"` para `ENUM("b", "a")`. A string vazia vem antes de strings não-vazias, e valores `NULL` vem antes de todos os outros valores de enumeração. Para evitar resultados inesperados, especifique a lista `ENUM` em ordem alfabética. Você também pode usar `GROUP BY CONCAT(col)` para ter certeza de que as colunas estão ordenadas alfabeticamente e não pelo índice numérico.

Se você quiser obter todos os valores possíveis para uma coluna `ENUM`, você deve usar: `SHOW COLUMNS FROM nome_tabela LIKE nome_coluna_enum` e analizar a definição de `ENUM` na segunda coluna.

6.2.3.4 O Tipo `SET`

Um `SET` é um objeto string que pode ter zero ou mais valores, cada um deve ser escolhido de uma lista de valores permitidos especificados quando a tabela é criada. Valores de colunas `SET` que consistem de múltiplos membros são espeficados separados por virgula (‘`,`’). Uma consquência distop é que valores dos membros de `SET` não podem, eles mesmos, conter vírgula.

Por exemplo, uma coluna especificada como `SET("um", "dois") NOT NULL` pode ter qualquer um destes valores:

```
""
"um"
"dois"
"um, dois"
```

Um `SET` pode ter no máximo 64 membros diferentes.

A partir da versão 3.23.51, espaços extras são automaticamente removidos dos valores de `SET` quando a tabela é criada.

MySQL armazena valores `SET` numericamente, com o bit de baixa-ordem do valor armazenado correspondendo ao primeiro membro do conjunto. Se você recupera um valor `SET` em um contexto numérico, o valor recuperado tem o conjunto de bits correspondente aos membros que aparecem no valor da coluna. Por exemplo, você pode recuperar valores numéricos de uma coluna `SET` assim:

```
mysql> SELECT col_set+0 FROM nome_tabela;
```

Se um número é armazenado em uma coluna `SET`, os bits que estão habilitados (com 1) na representação binária do número determinam o qual o membro no valor da coluna. Suponha uma coluna especificada como `SET("a","b","c","d")`. Então os membros terão os seguintes valores binários:

`SET` membro	Valor decimal	Valor binário
`a`	`1`	`0001`

```
b           2           0010
c           4           0100
d           8           1000
```

Se você atribuir um valor `9` a esta coluna, que é `1001` em binário, o primeiro e o quarto valores membros do `SET "a"` e `"d"` são selecionados e o valor resultante é `"a,d"`.

Para um valor contendo mais que um elemento de `SET`, não importa em qual ordem os elementos são listados quando foram inseridos seus valores. Também não importa quantas vezes um dado elemento e listado no valor. Quando o valor é recuperado posteriormente, cada elemento aparecerá uma vez, listados de acordo com a ordem em que eles foram especificados na crição da tabela. Por exemplo, se uma coluna é especificada como `SET("a","b","c","d")`, então `"a,d"`, `"d,a"` e `"d,a,a,d,d"` irão todos aparecer como `"a,d"` quando recuperados.

Se você define um valor que não é suportado pela coluna `SET`, o valor será ignorado.

Valores `SET` são ordenados numéricamente. Valores `NULL` vêm antes de valores `SET` não `NULL`.

Normalmente, você realiza um `SELECT` em uma coluna `SET` usando o operador `LIKE` ou a função `FIND_IN_SET()`:

```
mysql> SELECT * FROM nome_tabela WHERE col_set LIKE '%valor%';
mysql> SELECT * FROM nome_tabela WHERE FIND_IN_SET('valor',col_set)>0;
```

Mas o seguinte também funciona:

```
mysql> SELECT * FROM nome_tabela 2 WHERE col_set = 'val1,val2';
mysql> SELECT * FROM nome_tabela 3 WHERE col_set & 1;
```

A primeira desta instruções procura por uma correpondencia exata. A segunda por valores contendo o primeiro membro.

Se você quer obter todos os valores possíveis para uma coluna `SET`, você deve usar: `SHOW COLUMNS FROM nome_tabela LIKE nome_coluna_set` e analizar a definição do `SET` na segunda coluna.

6.2.4 Escolhendo o Tipo Correto para uma Coluna

Para um uso mais eficiente do armzenamento, tente usar o tipo mais adequado em todos os casos. Por exemplo, se um campo de inteiro for usado para valores em uma faixa entre `1` e `99999`, `MEDIUMINT UNSIGNED` é o melhor tipo.

Represtação precisa de valores monetários é um priblema comum. No MySQL você deve usar o tipo `DECIMAL`. Ele armazena uma string, então nenhuma perda de precisão deve ocorrer. Se a precisão não é tão importante, o tipo `DOUBLE` pode ser satisfatório.

Para uma alta precisão você sempre pode converter para um tipo de ponto fixo armazenado em um `BIGINT`. Isto perite fazer todos os cálculos com inteiros e converter o resultado para um ponto flutuante somente quando necessário.

6.2.5 Usando Tipos de Colunas de Outros Mecanismos de Banco de Dados

Para facilitar o uso de code para implementações SQL de outras empresas, MySQL mapeia os tipos de campos como mostrado na tabela seguinte. Este mapeamento torna fácil mudar definições de tabelas de outros mecanismos de banco de dados para o MySQL:

Tipo de outras empresas	Tipo MySQL
BINARY(NUM)	CHAR(NUM) BINARY
CHAR VARYING(NUM)	VARCHAR(NUM)
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT
LONG VARBINARY	MEDIUMBLOB
LONG VARCHAR	MEDIUMTEXT
MIDDLEINT	MEDIUMINT
VARBINARY(NUM)	VARCHAR(NUM) BINARY

O mapeamento do tipo de campo ocorre na criação da tabela. Se você cria uma tabela com tipos usador por outras empresas e então executa uma instrução `DESCRIBE nome_tabela`, MySQL relaciona a estrutura de tabela utilizando os tipos equivalentes do MySQL.

6.2.6 Exigências de Armazenamento dos Tipos de Coluna

As exigências de armazenamento para cada um dos tipos de colunas suportados pelo MySQL estão listados por categoria.

Exigências de armazenamento para tipos numéricos

Tipo da coluna	Tamanho exigido
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT	4 bytes
INTEGER	4 bytes
BIGINT	8 bytes
FLOAT(X)	4 se X <= 24 ou 8 se 25 <= X <= 53
FLOAT	4 bytes
DOUBLE	8 bytes
DOUBLE PRECISION	8 bytes
REAL	8 bytes
DECIMAL(M,D)	M+2 bytes se D > 0, M+1 bytes se D = 0 (D+2, se M < D)
NUMERIC(M,D)	M+2 bytes se D > 0, M+1 bytes se D = 0 (D+2, se M < D)

Exigência de armazenamento para tipos data e hora

Tipo de coluna	Tamanho exigido
DATE	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
TIME	3 bytes
YEAR	1 byte

Exigência de armazenamento para tipos string

Tipo de coluna	Tamanho exigido
CHAR(M)	M bytes, 1 <= M <= 255
VARCHAR(M)	L+1 bytes, onde L <= M e 1 <= M <= 255
TINYBLOB, TINYTEXT	L+1 bytes, onde L < 2^8
BLOB, TEXT	L+2 bytes, onde L < 2^16
MEDIUMBLOB, MEDIUMTEXT	L+3 bytes, onde L < 2^24
LONGBLOB, LONGTEXT	L+4 bytes, onde L < 2^32
ENUM('valor1','valor2',...)	1 ou 2 bytes, dependendo do número de valores enumerados (65535 valores no máximo)
SET('valor1','valor2',...)	1, 2, 3, 4 or 8 bytes, dependendo do número de membros do conjunto (64 membros no máximo)

Tipos VARCHAR, BLOB e TEXT são de tamanho variáveis, tendo o tamanho exigido para armazenamento dependendo do tamanho atual dos valores da coluna (representado por L na tabela anterior), e não do tamanho máximo do tipo. Por exemplo, uma coluna VARCHAR(10) pode guardar uma string com um tamanho máximo de 10 caracteres. O tamanho exigido para armazenamento atual é o tamanho da string (L), mais 1 byte para para gravar o tamanho da string. Por exemplo, para a string 'abcd', L é 4 e o tamanho exigido para armazenamento é 5 bytes.

Os tipos BLOB e TEXT exigem 1, 2, 3 ou 4 bytes para gravar o tamanho do valor da coluna, dependendo do tamanho máximo possível do tipo. Veja Seção 6.2.3.2 [BLOB], Página 498.

Se uma tabela inclui qualquer tipo de coluna de tamanho variável, o formato do registro também será de tamanho variável. Note que quando uma tabela é criada, MySQL pode, sob certas condições, mudar uma coluna de um tipo de tamanho variável para um tipo de tamanho fixo, ou vice-versa. Veja Seção 6.5.3.1 [Silent column changes], Página 607.

O tamanho de um objeto ENUM é determinado por um número de diferntes valores enumerados. Um byte é usado para enumerações até 255 valores possíveis. Dois bytes são usados para enumerações até 65535 valores. Veja Seção 6.2.3.3 [ENUM], Página 499.

O tamanho de uma objeto é determinado pelo número de diferentes membros do conjunto. Se o tamanho do conjunto é N, o objeto ocupa (N+7)/8 bytes, arredondados acima para 1, 2, 3, 4, ou 8 bytes. Um SET pode ter no máximo 64 membros. Veja Seção 6.2.3.4 [SET], Página 500.

O tamanho máximo de um registro em uma tabela MyISAM é 65534 bytes. Cada coluna BLOB e TEXT ocupa apenas 5-9 bytes deste tamanho.

6.3 Funções para Uso em Cláusulas SELECT e WHERE

Um `select_expression` ou `where_definition` em uma instrução SQL pode consistir de qualquer expressão utilizando as funções descritas abaixo.

Uma expressão que contém `NULL` sempre produz um valor `NULL` a menos que esteja indicado na dodumentação para os operandos e funções envolvidos na expressão.

Nota: Não deve haver nenhum espaço em branco entre um nome de função e os parentesis que a seguem. Isto ajuda o analizador MySQL a distinguir entre chamadas de funções e referências a tabelas ou colunas que possuem o mesmo nome de uma função. Espaços entre argumentos são permitidos.

Você pode forçar o MySQL a aceitar espaços depois do nome de funções iniciando o `mysqld` com a opção `--ansi` ou usando o `CLIENT_IGNORE_SPACE` no `mysql_connect()`, mas neste caso nome de funções se tornarão palavras reservadas. Veja Seção 1.8.2 [ANSI mode], Página 42.

Para sermos breve, exemplos mostram a saida do programa `mysql` na forma abreviada. Então isto:

```
mysql> SELECT MOD(29,9);
1 rows in set (0.00 sec)

+-----------+
| mod(29,9) |
+-----------+
|         2 |
+-----------+
```

é mostrado desta forma:

```
mysql> SELECT MOD(29,9);
        -> 2
```

6.3.1 Operadores e Funções de Tipos não Especificados

6.3.1.1 Parenteses

```
( ... )
```

Use parenteses para forçar a ordem em que as expressões serão avaliadas. Por exemplo:

```
mysql> SELECT 1+2*3;
        -> 7
mysql> SELECT (1+2)*3;
        -> 9
```

6.3.1.2 Operadores de Comparação

Operações de comparação resultam em um valor `1` (VERDADEIRO), `0` (FALSO), ou `NULL`. Estas funções funcionam tanto para tipos numéricos quanto para tipos strings. Strings

são convertidas automaticamente para números e números para strings quando necessário (como em Perl).

MySQL realiza comparações de acordo com as seguintes regras:

- Se um ou ambos os argumentos são `NULL`, o resultado da comparação é `NULL`, exceto para o operador `<=>`.
- Se ambos os argumentos em uma comparação são strings, eles são comparados como strings.
- Se ambos os argumentos são inteiros, eles são comparados como inteiros.
- Valores hexadecimais são tratados como strings binárias se não comparadas a um número.
- Se uma dos argumentos é uma coluna `TIMESTAMP` ou `DATETIME` e o outro argumento é uma constante, a constante é convertida para um timestamp antes da comparação ser realizada. Isto ocorre para ser mais amigável ao ODBC.
- Em todos os outros casos, os argumentos são coparados como números de ponto flutuante (real).

Por padrão, comparações de string são feita de modo independente do caso, usando o conjunto de caracteres atual (ISO-8859-1 Latin1 por padrão, o qual também funciona de forma excelente para o Inglês).

Se você está comparando strings em caso insensitivo com qualquer dos operadores padrões (`=`, `<>`..., mas não o `LIKE`) espaços em branco no fim da string (espaços, tabs e quebra de linha) serão ignorados.

```
mysql> SELECT "a" ="A \n";
        -> 1
```

Os seguintes exemplos ilustram a conversão de strings para números para operações de comparação:

```
mysql> SELECT 1 > '6x';
         -> 0
mysql> SELECT 7 > '6x';
         -> 1
mysql> SELECT 0 > 'x6';
         -> 0
mysql> SELECT 0 = 'x6';
         -> 1
```

Note que quando você está comparando uma coluna string com um número, o MySQL não pode usar índices para encontrar o valor rapidamente:

```
SELECT * FROM table_name WHERE string_key=1
```

A razão para isto é que existem muitas strings diferentes que podem retornar o valor `1`: `"1"`, `" 1"`, `"1a"` ...

`=` Igual:

```
mysql> SELECT 1 = 0;
        -> 0
mysql> SELECT '0' = 0;
```

```
        -> 1
mysql> SELECT '0.0' = 0;
        -> 1
mysql> SELECT '0.01' = 0;
        -> 0
mysql> SELECT '.01' = 0.01;
        -> 1
```

`<>`
`!=` Diferente:

```
mysql> SELECT '.01' <> '0.01';
        -> 1
mysql> SELECT .01 <> '0.01';
        -> 0
mysql> SELECT 'zapp' <> 'zappp';
        -> 1
```

`<=` Menor que ou igual:

```
mysql> SELECT 0.1 <= 2;
        -> 1
```

`<` Menor que:

```
mysql> SELECT 2 < 2;
        -> 0
```

`>=` Maior que ou igual:

```
mysql> SELECT 2 >= 2;
        -> 1
```

`>` Maior que:

```
mysql> SELECT 2 > 2;
        -> 0
```

`<=>` Igual para `NULL`:

```
mysql> SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;
        -> 1 1 0
```

`IS NULL`
`IS NOT NULL`

Teste para saber se um valor é ou não `NULL`:

```
mysql> SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;
        -> 0 0 1
mysql> SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;
        -> 1 1 0
```

Para estar apto a funcionar bem com outros programas, MySQL suporta os seguintes recursos extras quando utiliza-se `IS NULL`:

- Você pode encontrar o último registro inserido com:

  ```
  SELECT * FROM nome_tabela WHERE auto_col IS NULL
  ```

Isto pode ser desabilitado configurando `SQL_AUTO_IS_NULL=0`. Veja Seção 5.5.6 [SET OPTION], Página 460.

- Para colunas `DATE` e `DATETIME NOT NULL` você pode encontrar a data especial `0000-00-00` utilizando:

```
SELECT * FROM nome_tabela WHERE coluna_data IS NULL
```

Isto é necessário para que algums aplicações ODBC funcionem (já que ODBC não tem suporte a data `0000-00-00`)

`expr BETWEEN min AND max`

Se `expr` é maior que ou igual a `min` e `expr` é menor que ou igual a `max`, `BETWEEN` retorna `1`, senão é retornado `0`. Isto é equivalente a expressão `(min <= expr AND expr <= max)` se todos os argumentos são do mesmo tipo. Senão os tipos são convertidos, conforme as regras acima, e aplicadas a todos os três argumentos. **Note** que antes da versão 4.0.5 argumentos eram convertidos para o tipo da `expr`.

```
mysql> SELECT 1 BETWEEN 2 AND 3;
        -> 0
mysql> SELECT 'b' BETWEEN 'a' AND 'c';
        -> 1
mysql> SELECT 2 BETWEEN 2 AND '3';
        -> 1
mysql> SELECT 2 BETWEEN 2 AND 'x-3';
        -> 0
```

`expr NOT BETWEEN min AND max`

O mesmo que `NOT (expr BETWEEN min AND max)`.

`expr IN (valor,...)`

Retorna `1` se `expr` é qualquer dos valores na lista `IN`, senão retorna `0`. Se todos os valores são constantes, então os valores são avaliados de acordo com o tipo da `expr` e ordenado. A busca do item é então feita usando pesquisa binária. Isto significa que `IN` é muito rápido se os valores da lista `IN` forem todos contantes. Se `expr` é uma expressão strig em caso-sensitivo, a comparação é realizadas no modo caso-sensitvo:

```
mysql> SELECT 2 IN (0,3,5,'wefwf');
        -> 0
mysql> SELECT 'wefwf' IN (0,3,5,'wefwf');
        -> 1
```

O número de valores na lista `IN` é limitada apenas pelo valor `max_allowed_packet`.

Na versão 4.1 (para se adequar ao padrão SQL-99), `IN` returna `NULL` não apeans se a expressão a sua esquerda é `NULL`, mas também se nenhuma correspondência é encontrada na lista e uma de suas expressões é `NULL`.

A partir do MySQL versão 4.1, uma cláusula IN() também pode conter uma subquery. Veja Seção 6.4.2.3 [ANY IN SOME subqueries], Página 572.

`expr NOT IN (valor,...)`

O mesmo que `NOT (expr IN (valor,...))`.

`ISNULL(expr)`

Se `expr` é `NULL`, `ISNULL()` retorna `1`, senão retorna `0`:

```
mysql> SELECT ISNULL(1+1);
        -> 0
mysql> SELECT ISNULL(1/0);
        -> 1
```

Note que a compração de valores `NULL` usando `=` sempre será falso!

`COALESCE(lista)`

Retorna o primeiro elemento não `NULL` na lista:

```
mysql> SELECT COALESCE(NULL,1);
        -> 1
mysql> SELECT COALESCE(NULL,NULL,NULL);
        -> NULL
```

`INTERVAL(N,N1,N2,N3,...)`

Retorna `0` se `N < N1`, `1` se `N < N2` e assim por diante ou `-1` se `N` é `NULL`. Todos os argumentos são tratados como inteiros. Isto exige que `N1 < N2 < N3 < ... < Nn` para que esta função funcione corretamente. Isto ocorre devido a utilização pesquisa binária (muito rápida):

```
mysql> SELECT INTERVAL(23, 1, 15, 17, 30, 44, 200);
        -> 3
mysql> SELECT INTERVAL(10, 1, 10, 100, 1000);
        -> 2
mysql> SELECT INTERVAL(22, 23, 30, 44, 200);
        -> 0
```

6.3.1.3 Operadores Logicos

Em SQL, todos os operadores logicos avaliam TRUE (VERDADEIRO), FALSE (FALSO) ou NULL (DESCONHECIDO). No MySQL, esta implementação é como `1` (TRUE), `0` (FALSE), e `NULL`. A maioria deles é comum entre diferentes bancos de dados SQL. no entanto alguns podem retonar qualquer valor diferente de zero para TRUE.

`NOT`

`!`

NOT logico. Avalia como `1` se o operador é `0`, como `0` se o operador é diferente de zero, e `NOT NULL` retorna `NULL`.

```
mysql> SELECT NOT 10;
        -> 0
mysql> SELECT NOT 0;
        -> 1
mysql> SELECT NOT NULL;
        -> NULL
mysql> SELECT ! (1+1);
        -> 0
mysql> SELECT ! 1+1;
        -> 1
```

O último exemplo produz `1` pois a a expressão é avaliada como `(!1)+1`.

`AND`
`&&` AND lógico. Avalia como `1` se todos os operandos são diferentes de zero e não é `NULL`, como `0` se um ou mais operandos são `0`, senão retorna `NULL`.

```
mysql> SELECT 1 && 1;
        -> 1
mysql> SELECT 1 && 0;
        -> 0
mysql> SELECT 1 && NULL;
        -> NULL
mysql> SELECT 0 && NULL;
        -> 0
mysql> SELECT NULL && 0;
        -> 0
```

Por favor note que as versões do MySQL anteriores a versão 4.0.5 param a avaliação quando um valor `NULL` é encontrado, e não continua o processo buscando por possíveis `0`s. Isto significa que nessa versão, `SELECT (NULL AND 0)` retorna `NULL` ao invés de `0`. Na versão 4.0.5 o código tem sido re-elaborado para que o resultado sempre seja como prescrito pelo padrão SQL utilizando a otimização sempre que possível.

`OR`
`||` OR lógico. Avalia como `1` se algum operando é diferente de zero e como `NULL` se algum operando for `NULL`, senão `0` é retornado.

```
mysql> SELECT 1 || 1;
        -> 1
mysql> SELECT 1 || 0;
        -> 1
mysql> SELECT 0 || 0;
        -> 0
mysql> SELECT 0 || NULL;
        -> NULL
mysql> SELECT 1 || NULL;
        -> 1
```

`XOR` XOR lógico. Retorna `NULL` se o operando também é `NULL`. Para operandos não `NULL`, avalia como `1` se um número ímpar de operandos é diferente de zero, senão `0` é retornado.

```
mysql> SELECT 1 XOR 1;
        -> 0
mysql> SELECT 1 XOR 0;
        -> 1
mysql> SELECT 1 XOR NULL;
        -> NULL
mysql> SELECT 1 XOR 1 XOR 1;
        -> 1
```

`a XOR b` é matematicamente igual a `(a AND (NOT b)) OR ((NOT a) and b)`.

`XOR` foi adicionado na versão 4.0.2.

6.3.1.4 Funções de Fluxo de Controle

`CASE valor WHEN [valor comparado] THEN resultado [WHEN [valor comparado] THEN resultado ...] [ELSE resultado] END`
`CASE WHEN [condição] THEN result [WHEN [condição] THEN resultado ...] [ELSE resultado] END`

A primeira expressão retorna o `resultado` onde `valor=valor comparado`. A segunda expressão retorna o o resultado da primeira condição, a qual é verdadeira. Se não existe nenhum resultado correspondente, então o resultado depois do `ELSE` é retornado. Se não existe parte `ELSE` então é retornado `NULL` is returned:

```
mysql> SELECT CASE 1 WHEN 1 THEN "um"
           WHEN 2 THEN "dois" ELSE "mais" END;
       -> "one"
mysql> SELECT CASE WHEN 1>0 THEN "verdadeiro" ELSE "falso" END;
       -> "true"
mysql> SELECT CASE BINARY "B" WHEN "a" THEN 1 WHEN "b" THEN 2 END;
       -> NULL
```

O tipo do valor de retorno (`INTEGER`, `DOUBLE` ou `STRING`) é do mesmo tipo do primeiro valor retornado (a expressão depois do primeiro `THEN`).

`IF(expr1,expr2,expr3)`

Se `expr1` é VERDADEIRA (`expr1 <> 0` e `expr1 <> NULL`) então `IF()` retorna `expr2`, senão ela retorna `expr3`. `IF()` returna um valor numérico ou string, dependendo do contexto no qual é usado.

```
mysql> SELECT IF(1>2,2,3);
        -> 3
mysql> SELECT IF(1<2,'sim','não');
        -> 'sim'
mysql> SELECT IF(STRCMP('teste','teste1'),'não','sim');
        -> 'não'
```

Se `expr2` ou `expr3` é explicitamente `NULL` então o tipo resultante da função `IF()` é o tipo da coluna não `NULL`. (Este comportamento é novo na versão 4.0.3 do MySQL).

`expr1` é avaliada como um valor inteiro, o qual significa que se você está testando valores de ponto flutuante ou strings, você de fazê-lo usando um operando de comparação:

```
mysql> SELECT IF(0.1,1,0);
        -> 0
mysql> SELECT IF(0.1<>0,1,0);
        -> 1
```

No primeiro caso acima, `IF(0.1)` retorna `0` porque `0.1` é convertido para um valor inteiro, resultando um um teste `IF(0)`. Isto pode não ser o que você

esperava. No segundo caso, a comparação testa se o valor de ponto flutuante não é zero. O resultado da comparação converte o termo em um interiro.

O tipo de retorno padrão de `IF()` (o que pode importar quando ele é armazenado em uma tabela temporária) é calculado na versão 3.23 do MySQL de seguinte forma:

Expressão	**Valor de retorno**
expr2 ou expr3 retorna string	string
expr2 ou expr3 retorna um valor de ponto flutuante	ponto flutuante
expr2 ou expr3 retorna um inteiro	inteiro

Se expr2 e expr3 são strings, então o resultado é caso-insensitivo se ambas strings são caso insensitivo. (A patir da versão 3.23.51)

`IFNULL(expr1,expr2)`

Se `expr1` não é `NULL`, `IFNULL()` retorna `expr1`, senão retorna `expr2`. `IFNULL()` retorna um valor numérico ou string, dependendo do contexto no qual é usado:

```
mysql> SELECT IFNULL(1,0);
        -> 1
mysql> SELECT IFNULL(NULL,10);
        -> 10
mysql> SELECT IFNULL(1/0,10);
        -> 10
mysql> SELECT IFNULL(1/0,'yes');
        -> 'yes'
```

Na versão 4.0.6 e acima o valor resultante padrão de `IFNULL(expr1,expr2)` é o mais geral das duas expressões, na seguinte ordem: `STRING`, `REAL` ou `INTEGER`. A diferença das versões anteriores é mais notável quando se cria uma tabela baseada em uma expressão ou o MySQL tem que armazenar internamente um valor de `IFNULL()` em uma tabela temporária.

```
CREATE TABLE foo SELECT IFNULL(1,"teste") as teste;
```

Na versão 4.0.6 do MySQL o tipo da coluna 'teste' é `CHAR(4)` enquanto nas versões anteriores ela seria do tipo `BIGINT`.

`NULLIF(expr1,expr2)`

Se `expr1 = expr2` for verdadeiro, é retornado `NULL` senão é retornado `expr1`. Isto é o mesmo que `CASE WHEN x = y THEN NULL ELSE x END`:

```
mysql> SELECT NULLIF(1,1);
        -> NULL
mysql> SELECT NULLIF(1,2);
        -> 1
```

Note que `expr1` é avaliada duas vezes no MySQL se os argumentos não são iguais.

6.3.2 Funções String

Funções string retornam `NULL` se o tamanho do resultado for maior que o parâmetro do servidor `max_allowed_packet`. Veja Seção 5.5.2 [Server parameters], Página 454.

Para funções que operam com as posições de uma string, a primeira posição é numerada como 1.

`ASCII(str)`
: Retorna o valor do código ASCII do caracter mais a esquerda da string `str`. Retorna `0` se `str` é uma string vazia. Retorna `NULL` se `str` é `NULL`:

```
mysql> SELECT ASCII('2');
        -> 50
mysql> SELECT ASCII(2);
        -> 50
mysql> SELECT ASCII('dx');
        -> 100
```

Veja também a função `ORD()`.

`BIN(N)`
: Retorna um representação string do valor binário de `N`, onde `N` é um número muito grande (`BIGINT`). Isto é equivalente a `CONV(N,10,2)`. Retorna `NULL` se `N` é `NULL`:

```
mysql> SELECT BIN(12);
        -> '1100'
```

`BIT_LENGTH(str)`
: Retorna o tamanho da string `str` em bits:

```
mysql> SELECT BIT_LENGTH('text');
        -> 32
```

`CHAR(N,...)`
: `CHAR()` interpretia os argumentos como inteiros e retorna uma string com caracteres dados pelo valor do código ASCII referentes a estes inteiros. Valores `NULL` são desconsiderados:

```
mysql> SELECT CHAR(77,121,83,81,'76');
        -> 'MySQL'
mysql> SELECT CHAR(77,77.3,'77.3');
        -> 'MMM'
```

`CONCAT(str1,str2,...)`
: Retorna a string resultante da concatenação dos argumentos. Retorna `NULL` se qualquer dos argumentos for `NULL`. Pode ter mais de 2 argumentos. Um argumento numérico é convertido para sua forma string equivalente:

```
mysql> SELECT CONCAT('My', 'S', 'QL');
        -> 'MySQL'
mysql> SELECT CONCAT('My', NULL, 'QL');
        -> NULL
mysql> SELECT CONCAT(14.3);
        -> '14.3'
```

`CONCAT_WS(separador, str1, str2,...)`

`CONCAT_WS()` significa CONCAT With Separator (CONCAT com separador) e é uma forma especial do `CONCAT()`. O primeiro argumento é o separador para os outros argumentos. O separador é adicionado entre as strings a serem concatenadas: O separador pode ser uma string assim como os outros argumentos. Se o separador é `NULL`, o resultado será `NULL`. A função irá desconsiderar qualquer `NULL` depois do argumento do separador.

```
mysql> SELECT CONCAT_WS(",","First name","Second name","Last Name");
       -> 'First name,Second name,Last Name'
mysql> SELECT CONCAT_WS(",","First name",NULL,"Last Name");
       -> 'First name,Last Name'
```

Antes do MySQL 4.1.1, `CONCAT_WS()` desconsiderava strings vazias assim como valores `NULL`.

`CONV(N,da_base,para_base)`

Converte números entre diferentes bases. Retorna uma representação string do número `N`, convertido da base `da_base` para base `para_base`. Retorna `NULL` se qualquer argumento é `NULL`. O argumento `N` é interpretado como um inteiro, mas pode ser especificado como um inteiro ou uma string. A base mínima é `2` e a máxima é `36`. Se `para_base` é um número negativo, `N` é considerado como um número com sinal. Caso contrário, `N` é tratado como um número sem sinal. `CONV` funciona com precisão de 64-bit:

```
mysql> SELECT CONV("a",16,2);
        -> '1010'
mysql> SELECT CONV("6E",18,8);
        -> '172'
mysql> SELECT CONV(-17,10,-18);
        -> '-H'
mysql> SELECT CONV(10+"10"+'10'+0xa,10,10);
        -> '40'
```

`ELT(N,str1,str2,str3,...)`

Retorna `str1` se `N = 1`, `str2` se `N = 2`, e assim por diante. Retorna `NULL` se `N` é menor que `1` ou maior que o número de argumentos. `ELT()` é o complemento de `FIELD()`:

```
mysql> SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');
        -> 'ej'
mysql> SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');
        -> 'foo'
```

`EXPORT_SET(bits,on,off,[separador,[numero_de_bits]])`

Retorna uma string onde para todo bit 1 em 'bit', você obtém uma string 'on' e para cada bit 0 você obtem uma string 'off', Cada string é separada com 'separador' (padrão,',') e só 'número_de_bits' (padrão 64) de 'bits' é usado:

```
mysql> SELECT EXPORT_SET(5,'S','N',',',4)
        -> S,N,S,N
```

`FIELD(str,str1,str2,str3,...)`

Retorna o índice de `str` na lista `str1`, `str2`, `str3`, ... Retorns `0` se `str` não for encontrada. `FIELD()` é o complemento de `ELT()`:

```
mysql> SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');
        -> 2
mysql> SELECT FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');
        -> 0
```

`FIND_IN_SET(str,strlista)`

Retorna um valor `1` para `N` se a string `str` está na lista `strlist` contendo `N` substrings. A lista de string é composta de substrings separadas pelo caracter '`,`'. Se o primeiro argumento é uma string constante e o segundo é uma coluna do tipo `SET`, a função `FIND_IN_SET()` é otimizada para usar aritmética binária! Retorna `0` se `str` não está na `strlista` ou se `strlista` é uma string vazia. Retorna `NULL` se os argumentos são `NULL`. Esta função não irá funcionar adequadamente se o primeiro argumento contém uma vírgula ('`,`'):

```
mysql> SELECT FIND_IN_SET('b','a,b,c,d');
        -> 2
```

`HEX(N_ou_S)`

Se N_OU_S é um número, é retornado um representação string do valor hexadecimal de `N`, onde `N` é um número muito grande (`BIGINT`). Isto é equivalente a `CONV(N,10,16)`.

Se N_OU_S é uma string, é retornado uma string hexadecimal de N_OU_S onde cada caracter de N_OU_S é convertido para 2 dígitos hexadecimais. Isto é o inverso da string `0xff`.

```
mysql> SELECT HEX(255);
        -> 'FF'
mysql> SELECT HEX("abc");
        -> 616263
mysql> SELECT 0x616263;
        -> "abc"
```

`INSTR(str,substr)`

Retorna a posição da primeira ocorrência da substring `substr` na string `str`. É o mesmo que as o `LOCATE()` com dois argumentos, exceto pelo fato de que os argumentos estão tracados:

```
mysql> SELECT INSTR('foobarbar', 'bar');
        -> 4
mysql> SELECT INSTR('xbar', 'foobar');
        -> 0
```

Esta função é multi-byte. Na versão 3.23 do MySQL esta função é caso sensitivo, enquanto na versão 4.0 ela só é caso-sensitivo se os argumentos são uma string binária.

`INSERT(str,pos,tam,novastr)`

Retorna a string `str`, com a a substring começando na posição `pos` e contendo `tam` caracteres substituida pela string `novastr`:

```
mysql> SELECT INSERT('Quadratico', 3, 4, 'Onde');
        -> 'QuOndetico'
```

Esta função é multi-byte.

`LCASE(str)`
`LOWER(str)`

Retorna a string `str` com todos caracteres alterados para letra minúsculas de acordo com o conjunto de caracteres atual (o padrão é ISO-8859-1 Latin1):

```
mysql> SELECT LCASE('MYSQL');
        -> 'mysql'
```

Esta é uma função multi-byte.

`LEFT(str,tam)`

Retorna os `tam` caracteres mais a esquerda da string `str`:

```
mysql> SELECT LEFT('foobarbar', 5);
        -> 'fooba'
```

Esta função é multi-byte.

`LOAD_FILE(nome_arquivo)`

Lêb o arquivo e retona o conteudo do arquivo como uma string. O arquivo beve estar no servidor, você deve especificar o caminho completo para o arquivo, e você deve ter o privilégio `FILE`. O arquivo deve ser legível para todos e ser menor que o especificado em `max_allowed_packet`.

Se o arquivo não existe ou não pode ser lido devido a alguma das razões acima, a função retornará `NULL`:

```
mysql> UPDATE nome_tabela
           SET coluna_blob=LOAD_FILE("/tmp/picture")
           WHERE id=1;
```

Se você não está usando a versão 3.23 MySQL, você tem que fazer a leitura do arquivo dentro do seu aplicativo e criar uma instrução `INSERT` para atualizar o banco de dados com a informação do arquivo. Um modo de se fazer isto, se você estiver usando a biblioteca MySQL++, pode ser encontrada em `http://www.mysql.com/documentation/mysql++/mysql++-examples.html`.

`LOCATE(substr,str)`
`LOCATE(substr,str,pos)`

A primeira sintaxe retorna a posição da primeira ocorrência da substring `substr` na string `str`. A segunda sintaxe retorna a posição da primeira ocorrência da substring `substr` na string `str`, iniciando na posição `pos`. Retornam `0` se `substr` não está em `str`:

```
mysql> SELECT LOCATE('bar', 'foobarbar');
        -> 4
mysql> SELECT LOCATE('xbar', 'foobar');
        -> 0
mysql> SELECT LPAD('hi',4,'??');
        -> '??hi'
```

`LTRIM(str)`

Retorna a string `str` com caracteres de espaços extras iniciais removidos:

```
mysql> SELECT LTRIM('  barbar');
        -> 'barbar'
```

`MAKE_SET(bits,str1,str2,...)`

Retorna um conjunto (uma string contendo substrings separadas por '`,`') contendo as strings que tem o bit correspondente em `bits` definido . `str1` corresponde ao bit 1, `str2` ao bit 2, etc. Strings `NULL` em `str1`, `str2`, `...` não são adicionadas ao resultado:

```
mysql> SELECT MAKE_SET(1,'a','b','c');
        -> 'a'
mysql> SELECT MAKE_SET(1 | 4,'Oi','meu','mundo');
        -> 'Oi,mundo'
mysql> SELECT MAKE_SET(0,'a','b','c');
        -> ''
```

`OCT(N)`

Retorna uma representação string do valor octal de `N`, onde `N` é um número muito grande. Isto é equivalente a `CONV(N,10,8)`. Retorna `NULL` se `N` é `NULL`:

```
mysql> SELECT OCT(12);
        -> '14'
```

`ORD(str)` Se o caracter mais a esquerda da string `str` é um caracter multi-byte, é retornado o código para este caracter, calculado a partir dos valores do código ASCII dos seus caracteres contituintes utizando-se a seguinte fórmula: `((primeiro byte do código ASCII)*256+(segundo byte do código ASCII))[*256+terceiro byte do código ASCII...]`. Se o caracter mais a esquerda não é multi-byte, é retornado o mesmo valor que a função `ASCII()` retorna:

```
mysql> SELECT ORD('2');
        -> 50
```

`LENGTH(str)`
`OCTET_LENGTH(str)`
`CHAR_LENGTH(str)`
`CHARACTER_LENGTH(str)`

Retorna o tamanho da string `str`:

```
mysql> SELECT LENGTH('text');
        -> 4
mysql> SELECT OCTET_LENGTH('text');
        -> 4
```

`LENGTH()` e `OCTET_LENGTH()` são sinônimos e medem o tamanho da length em bytes (octets). Um caracter multi-byte conta é considerado vários bytes. `CHAR_LENGTH()` e `CHARACTER_LENGTH()` são sinônimos e medem o tamanho da string em caracteres. Um caracter multi-byte conta como um único caracter. Isto significa que para uma string contendo cinco caracteres de dois bytes, `LENGTH()` retorna `10`, enquanto `CHAR_LENGTH()` retorna `5`.

`MID(str,pos,len)`

`MID(str,pos,len)` é um sinônimo para `SUBSTRING(str,pos,len)`.

`POSITION(substr IN str)`

`POSITION(substr IN str)` é um sinônimo para `LOCATE(substr,str)`.

`QUOTE(str)`

Coloca uma string entre aspas para produzir um resultado que possa ser usada em uma intrução SQL como um valor de dados com o caracter de escape correto. A string é retornada entre aspas simples e cada instâqncia de aspas simples (‘’’), barra invertida (‘\’), ASCII NUL, e Control-Z é precedida por uma barra invertida. Se o argumento é `NULL`, o valor retornado é a palavra “NULL” sem aspas simples.

A função `QUOTE()` foi adicionada na versão 4.0.3 do MySQL.

```
mysql> SELECT QUOTE("Don't");
        -> 'Don\'t!'
mysql> SELECT QUOTE(NULL);
        -> NULL
```

`REPEAT(str,cont)`

Retorna uma string consistindo da string `str` repetida `cont` vezes. Se `cont <= 0`, é retornado uma string vazia. É retornado `NULL` se `str` ou `cont` são `NULL`:

```
mysql> SELECT REPEAT('MySQL', 3);
        -> 'MySQLMySQLMySQL'
```

`REPLACE(str,da_str,para_str)`

Retorna a string `str` com todas ocorrências da string `da_str` substituida pela string `para_str`:

```
mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');
        -> 'WwWwWw.mysql.com'
```

Esta função é multi-byte.

`REVERSE(str)`

Returns the string `str` with the order of the characters reversed:

```
mysql> SELECT REVERSE('abc');
        -> 'cba'
```

Esta função é multi-byte.

`RIGHT(str,tem)`

```
mysql> SELECT RIGHT('foobarbar', 4);
        -> 'rbar'
```

Esta função é multi-byte.

`RPAD(str,tam,strpreech)`

Retorna a string `str`, preenchida a direita com a string `strpreench` para um tamanho de `tam` caracteres. Se `str` é maior que `tam`, o valor retornado é reduzido para `tam` caracteres.

```
mysql> SELECT RPAD('hi',5,'?');
        -> 'hi???'
```

`RTRIM(str)`

Retourna a string `str` com caracteres de espaços extras finais removidos:

```
mysql> SELECT RTRIM('barbar   ');
        -> 'barbar'
```

Esta função é multi-byte.

`SOUNDEX(str)`

Retorna uma string 'soundex' de `str`. Duas strings que parecidas fonéticamentea devem ter strings 'soundex' iguais. Uma string soundex padrão possui 4 caracteres, mas a função `SOUNDEX()` retorna uma string de tamanho arbitrário. Você posde usar `SUBSTRING()` no resultado para obter uma string 'soundex' padrão. Todos os caracteres não alfanuméricos são ignorados na string dada. Todas caracteres internacionais fora da faixa A-Z são tratados como vogais:

```
mysql> SELECT SOUNDEX('Hello');
        -> 'H400'
mysql> SELECT SOUNDEX('Quadratically');
        -> 'Q36324'
```

`SPACE(N)` Retorna uma string contendo `N` caracteres de espaço:

```
mysql> SELECT SPACE(6);
        -> '      '
```

`SUBSTRING(str,pos)`
`SUBSTRING(str FROM pos)`
`SUBSTRING(str,pos,tam)`
`SUBSTRING(str FROM pos FOR tam)`

A forma sem um argumento `tam` retorna uma substring da string `str` iniciando na posição `pos`. A forma com um argumento `tam` retorna a substring com `tam` caracteres da string `str`, iniciando da posição `pos`. A forma variante que utiliza `FROM` é a sintaxe SQL-92:

```
mysql> SELECT SUBSTRING('Quadratically',5);
        -> 'ratically'
mysql> SELECT SUBSTRING('foobarbar' FROM 4);
        -> 'barbar'
mysql> SELECT SUBSTRING('Quadratically',5,6);
        -> 'ratica'
```

Esta função é multi-byte.

`SUBSTRING_INDEX(str,delim,cont)`

Retorna a substring da string `str` antes de `cont` ocorrencias do delimitador `delim`. Se `cont` é positivo, tudo a esquerda do delimitador final (contando a partir da esquerda) é retornado. Se `cont` é negativo, tudo a direita do delimitador final (contando a partir da direita) é retornado.

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);
        -> 'www.mysql'
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);
        -> 'mysql.com'
```

Esta função é multi-byte.

`TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)`

Retorna a string `str` com todos prefixos e/ou sufixos `remstr` removidos. Se nenhum dos especificadores `BOTH`, `LEADING` ou `TRAILING` são dados, é considerado `BOTH`. Se `remstr` não é especificada, espaços são removidos:

```
mysql> SELECT TRIM('  bar   ');
        -> 'bar'
mysql> SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');
        -> 'barxxx'
mysql> SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');
        -> 'bar'
mysql> SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');
        -> 'barx'
```

Esta função é multi-byte.

`UCASE(str)`
`UPPER(str)`

Retorna a string `str` com todos caracteres alterados para letra maiúsculas de acordo com o conjunto de caracteres atual (o padrão é ISO-8859-1 Latin1):

```
mysql> SELECT UCASE('Hej');
        -> 'HEJ'
```

Esta é uma função multi-byte.

6.3.2.1 Funções de Comparação de Strings

MySQL automaticamente converte números para quando necessário, e vice-versa:

```
mysql> SELECT 1+"1";
        -> 2
mysql> SELECT CONCAT(2,' test');
        -> '2 test'
```

Se você quiser converter um número em uma string de forma explicita, passe-o como um argumento de `CONCAT()`.

Se uma função de string tem uma string binária como argumento, a string resultante é também um string binária. Um número convertido para uma string é tratado como um string binária. Isto afeta apenas a comparação.

Normalmente, se qualquer expressão em uma string é caso-sensitivo, a comparação é realizada no modo caso sensitivo.

`expr LIKE pad [ESCAPE 'car-escape']`

Correspondência de padrões usando uma simples expressão de comparações SQL. Retorna `1` (VERDADEIRO) ou `0` (FALSO). Com `LIKE` você pode usar os seguintes meta-caracteres no padrao:

Car	Descrição
%	Corresponde a qualquer número de caracteres, até zero caracteres
_	Corresponde a exatamente um caracter

```
mysql> SELECT 'David!' LIKE 'David_';
        -> 1
mysql> SELECT 'David!' LIKE '%D%v%';
        -> 1
```

Para testar instâncias literais de um meta caracter, preceda o caracter com o carcter de escape. Se você não especificar o caracter de `ESCAPE`, assume-se '\':

String	Description
\%	Correponde a um caracter %
_	Correponde a um caracter _

```
mysql> SELECT 'David!' LIKE 'David\_';
        -> 0
mysql> SELECT 'David_' LIKE 'David\_';
        -> 1
```

Para especificar um caracter de escape diferebte, use a cláusula `ESCAPE`:

```
mysql> SELECT 'David_' LIKE 'David|_' ESCAPE '|';
        -> 1
```

As seguintes instruções mostram que a comparação de strings são caso-insensitivo, a menos que um dos operandos seja uma string binária:

```
mysql> SELECT 'abc' LIKE 'ABC';
        -> 1
mysql> SELECT 'abc' LIKE BINARY 'ABC';
        -> 0
```

`LIKE` é permitido em uma expressão numérica! (Esta é uma extensão MySQL para o `LIKE` do SQL-99.)

```
mysql> SELECT 10 LIKE '1%';
        -> 1
```

Nota: Como MySQL usa sintaxe de escape do C em strings (por exemplo, '`\n`'), você deve dobrar qualquer '\' que você usar em sua string `LIKE`. Por exemplo, para pesquisar por '`\n`', especifique-o como '`\\n`'. Para buscar por '\', especifique-o como '`\\\\`' (as barras invertidas são eliminadas uma vez pelo analizador e outra vez quando a correspondência de padrões é feita, deixando uma únicas barra invertida para ser verificada).

Note: O `LIKE` atual não é um `caracter multi-byte`. Comparaçãoes são feitas caracter por caracter.

`expr NOT LIKE pad [ESCAPE 'car-escape']`
: O mesmo que `NOT (expr LIKE pad [ESCAPE 'car-escape'])`.

`expr SOUNDS LIKE expr`
: O mesmo que `SOUNDEX(expr)=SOUNDEX(expr)` (disponível apenas na versão 4.1 ou posterior).

`expr REGEXP pad`
`expr RLIKE pad`
: Realiza a busca de padrões em uma expressã string com base no padrão `pad`. O padrão pode ser uma expressão regular extendida. Veja Apêndice G [Regexp],

Página 1093. Retorna `1` se `expr` conincide com `pad`, senão retorna `0`. `RLIKE` é um sinônimo para `REGEXP`, fornecido para compatibilidade com `mSQL`. Nota: Como MySQL usa a sintaxe de escape do C em strings (por exemplo, '`\n`'), você deve dobrar qualquer '`\`' que você use em sua string `REGEXP`. Como na versão 3.23.4 do MySQL, `REGEXP` é caso- insensitivo para strings normais (não binárias).

```
mysql> SELECT 'Monty!' REGEXP 'm%y%%';
        -> 0
mysql> SELECT 'Monty!' REGEXP '.*';
        -> 1
mysql> SELECT 'new*\n*line' REGEXP 'new\\*.\\*line';
        -> 1
mysql> SELECT "a" REGEXP "A", "a" REGEXP BINARY "A";
        -> 1  0
mysql> SELECT "a" REGEXP "^[a-d]";
        -> 1
```

`REGEXP` e `RLIKE` usam o conjunto de caracteres atual (ISO-8859-1 Latin1 por padrão) para decidir o tipo de caracter.

`expr NOT REGEXP pad`
`expr NOT RLIKE pad`
: O mesmo que `NOT (expr REGEXP pad)`.

`STRCMP(expr1,expr2)`
: `STRCMP()` retorna `0` se as string são a mesma, `-1` se o primeiro argumento é menor que o segundo de acordo com a ordenação atual e `1` em caso contrário:

```
mysql> SELECT STRCMP('texto', 'texto2');
        -> -1
mysql> SELECT STRCMP('texto2', 'texto');
        -> 1
mysql> SELECT STRCMP('texto', 'texto');
        -> 0
```

`MATCH (col1,col2,...) AGAINST (expr [IN BOOLEAN MODE | WITH QUERY EXPANSION] )`
: `MATCH ... AGAINST()` é usado para busca de textos completos e retorna a relvância - similaridade medidad entre o texto nas colunas `(col1,col2,...)` e a consulta `expr`. Relevância é um número de ponto flutuante. Relevância zero significa que não houve nenhuma similaridade. `MATCH ... AGAINST()` está disponível na versão 3.23.23 ou posterior do MySQL. A extensão `IN BOOLEAN MODE` foi adicionada na versão 4.0.1, `WITH QUERY EXPANSION` foi adicionado na versão 4.1.1. Para detalhes e exemplos de uso, veja Seção 6.8 [Fulltext Search], Página 619.

6.3.2.2 Caso Sensitivo

`BINARY`
: O operador `BINARY` transforma uma string em uma string binária. Este é um modo fácil de forçar a comparação para se caso-sensitivo mesmo se a coluna não seja definida como `BINARY` ou `BLOB`:

```
mysql> SELECT "a" = "A";
        -> 1
mysql> SELECT BINARY "a" = "A";
        -> 0
```

`BINARY string` é um atalho para `CAST(string AS BINARY)`. Veja Seção 6.3.5 [Cast Functions], Página 544. `BINARY` foi introduzida na versão 3.23.0 do MySQL.

Note que em alguns contextos MySQL não estará apto a usar o índice de forma eficiente quando se transformar uma coluna índice em `BINARY`.

Se você quiser compara um blob caso-insensitivo você pode sempre convertê-lo para letras maiúsculas antes de faer a comparação:

```
SELECT 'A' LIKE UPPER(col_blobl) FROM nome_tabela;
```

Não planejamos introduzir em breve coerção (casting) entre diferentes conjuntos de caracteres para tornar comparções de strings mais flexível.

6.3.3 Funções Numéricas

6.3.3.1 Operações Aritiméticas

Os operadores aritiméticos usuais estão disponíveis. '`-`', '`+`', e '`*`', o resultado é calculado com precisão de `BIGINT` (64-bit) se ambos os argumentos são inteiros! Se um dos argumentos for um inteiro sem sinal, e o outro argumento é um inteiro também, o resultado será um inteiro sem sinal. Veja Seção 6.3.5 [Cast Functions], Página 544.

`+` Adição:

```
mysql> SELECT 3+5;
        -> 8
```

`-` Subtração:

```
mysql> SELECT 3-5;
        -> -2
```

`*` Multiplicação:

```
mysql> SELECT 3*5;
        -> 15
mysql> SELECT 18014398509481984*18014398509481984.0;
        -> 324518553658426726783156020576256.0
mysql> SELECT 18014398509481984*18014398509481984;
        -> 0
```

O resultado da última expressão é incorreta porque o resultado da multiplicação de inteiros excede a faixa de 64-bits dos cálculos `BIGINT`.

`/` Divisão:

```
mysql> SELECT 3/5;
        -> 0.60
```

Divisões por zero produz um resultado `NULL`:

```
mysql> SELECT 102/(1-1);
        -> NULL
```

Uma divisão será calculada com aritimética `BIGINT` somente se executada em um contexto no qual o resultado é convertido para um interiro!

6.3.3.2 Funções Matematicas

Todas as funções matematicas retornam `NULL` no caso de um erro.

`-` Menos unario. Muda o sinal do argumento:

```
mysql> SELECT - 2;
        -> -2
```

Note que se este operador é utilizando com um `BIGINT`, o valor retornado é um `BIGINT`! Isto significa que você deve evitar usar `-` em inteiros que pode ter o valor de `-2^63`!

`ABS(X)` Retorna o valor absoluto de `X`:

```
mysql> SELECT ABS(2);
        -> 2
mysql> SELECT ABS(-32);
        -> 32
```

O uso desta função é seguro com valores `BIGINT`.

`SIGN(X)` Retorna o sinal do argumento como `-1`, `0`, ou `1`, dependendo de quando `X` é negativo, zero, ou positivo:

```
mysql> SELECT SIGN(-32);
        -> -1
mysql> SELECT SIGN(0);
        -> 0
mysql> SELECT SIGN(234);
        -> 1
```

`MOD(N,M)`
`%` Modulo (como o operador `%` em C). Retorna o resto de `N` dividido por `M`:

```
mysql> SELECT MOD(234, 10);
        -> 4
mysql> SELECT 253 % 7;
        -> 1
mysql> SELECT MOD(29,9);
        -> 2
mysql> SELECT 29 MOD 9;
        -> 2
```

O uso desta função é seguro com valores `BIGINT`. O último exemplo só funciona no MySQL 4.1

`FLOOR(X)` Retorna o maior valor inteiro não maior que `X`:

```
mysql> SELECT FLOOR(1.23);
        -> 1
mysql> SELECT FLOOR(-1.23);
        -> -2
```

Note que o valor retornado é convertido para um `BIGINT`!

`CEILING(X)`
`CEIL(X)` Retorna o menor valor inteiro não menor que `X`:

```
mysql> SELECT CEILING(1.23);
        -> 2
mysql> SELECT CEIL(-1.23);
        -> -1
```

O alias `CEIL()` foi adicionado versão 4.0.6.

Note que o valor retornado é convertido para um `BIGINT`!

`ROUND(X)`
`ROUND(X,D)`

Retorna o argumeto `X`, arredondado para o inteiro mais próximo. Com dois argumentos o arredandamento é feito para um número com `D` decimais.

```
mysql> SELECT ROUND(-1.23);
        -> -1
mysql> SELECT ROUND(-1.58);
        -> -2
mysql> SELECT ROUND(1.58);
        -> 2
mysql> SELECT ROUND(1.298, 1);
        -> 1.3
mysql> SELECT ROUND(1.298, 0);
        -> 1
mysql> SELECT ROUND(23.298, -1);
        -> 20
```

Note que o comportamento de `ROUND()` quando o argumento está no meio do caminho entre dois inteiros depende da implementação da biblioteca C. Alguns arredondamentos para o número mais próximo, são sempre para baixo, para cima ou são zero. Se você precisa de um tipo de arredondamento, você deve usar uma função bem definida como `TRUNCATE()` ou `FLOOR()`.

`DIV` Divisão de inteiros. Similar ao `FLOOR()` mas seguro com valores `BIGINT`.

```
mysql> SELECT 5 DIV 2
-> 2
```

`DIV` é novo no MySQL 4.1.0.

`EXP(X)` Retorna o valor de `e` (the base of natural logarithms) raised to the power of `X`:

```
mysql> SELECT EXP(2);
        -> 7.389056
mysql> SELECT EXP(-2);
        -> 0.135335
```

`LN(X)` Retorna o logaritmo natural de `X`:

```
mysql> SELECT LN(2);
        -> 0.693147
mysql> SELECT LN(-2);
        -> NULL
```

Esta função foi adicionada na versão 4.0.3 do MySQL. É sinônimo de `LOG(X)` no MySQL.

`LOG(X)`
`LOG(B,X)` Se chamado com um parâmetro, esta função retorna o logarítmo natural de `X`:

```
mysql> SELECT LOG(2);
        -> 0.693147
mysql> SELECT LOG(-2);
        -> NULL
```

Se chamado com dois parâmetros, esta função retorna o logarítmo natural de `X` para uma base arbitraria `B`:

```
mysql> SELECT LOG(2,65536);
        -> 16.000000
mysql> SELECT LOG(1,100);
        -> NULL
```

A opção de base arbitrária foi adicionada na versão 4.0.3 do MySQL. `LOG(B,X)` é equivalente a `LOG(X)/LOG(B)`.

`LOG2(X)` Returna o logarítmo na base 2 de `X`:

```
mysql> SELECT LOG2(65536);
        -> 16.000000
mysql> SELECT LOG2(-100);
        -> NULL
```

`LOG2()` é útil para descobrir quantos bits um número necessitaria para ser armazenado. Esta função foi adicionada na versão 4.0.3 do MySQL. Em versões anteriores, você pode usar `LOG(X)/LOG(2)`.

`LOG10(X)` Returna o logarítmo na base 10 de `X`:

```
mysql> SELECT LOG10(2);
        -> 0.301030
mysql> SELECT LOG10(100);
        -> 2.000000
mysql> SELECT LOG10(-100);
        -> NULL
```

`POW(X,Y)`
`POWER(X,Y)`
Retorna o valor de `X` elevado a potência de `Y`:

```
mysql> SELECT POW(2,2);
        -> 4.000000
mysql> SELECT POW(2,-2);
        -> 0.250000
```

`SQRT(X)` Retorna o a raiz quadrada não negativa de `X`:

```
mysql> SELECT SQRT(4);
        -> 2.000000
mysql> SELECT SQRT(20);
        -> 4.472136
```

`PI()` Retorna o valor de PI. A quantidade de números decimais padrão é 5, mas o MySQL usa internamente a precisão dupla completa para PI.

```
mysql> SELECT PI();
        -> 3.141593
mysql> SELECT PI()+0.000000000000000000;
        -> 3.141592653589793116
```

`COS(X)` Retorna o cosseno de `X`, onde `X` é dado em radianos:

```
mysql> SELECT COS(PI());
        -> -1.000000
```

`SIN(X)` Retorna o seno de `X`, onde `X` é dado em radianos:

```
mysql> SELECT SIN(PI());
        -> 0.000000
```

`TAN(X)` Retorna a tangente de `X`, onde `X` é dado em radianos:

```
mysql> SELECT TAN(PI()+1);
        -> 1.557408
```

`ACOS(X)` Retorna o arco cosseno `X`, isto é, o valor cujo cosseno é `X`. Retorna `NULL` se `X` não está na faixa de `-1` a `1`:

```
mysql> SELECT ACOS(1);
        -> 0.000000
mysql> SELECT ACOS(1.0001);
        -> NULL
mysql> SELECT ACOS(0);
        -> 1.570796
```

`ASIN(X)` Retorna o arco seno `X`, isto é, o valor cujo seno é `X`. Retorna `NULL` se `X` não está na faixa de `-1` a `1`:

```
mysql> SELECT ASIN(0.2);
        -> 0.201358
mysql> SELECT ASIN('foo');
        -> 0.000000
```

`ATAN(X)` Retorna o arco tangente `X`, isto é, o valor cuja tangente é `X`. `X`:

```
mysql> SELECT ATAN(2);
        -> 1.107149
mysql> SELECT ATAN(-2);
        -> -1.107149
```

`ATAN(Y,X)`
`ATAN2(Y,X)`

Retorna o arco tangente de duas variaveis `X` e `Y`. É similar ao caclculo do arco tengente de `Y / X`, exceto que os sinais de ambos argumentos são usados para determinas o quadrante do resultado:

```
mysql> SELECT ATAN(-2,2);
        -> -0.785398
mysql> SELECT ATAN2(PI(),0);
        -> 1.570796
```

`COT(X)` Returns a cotangente de `X`:

```
mysql> SELECT COT(12);
        -> -1.57267341
mysql> SELECT COT(0);
        -> NULL
```

`CRC32(expr)`

Calcula um valor de verificação de redundância cíclica e retorna um valor unsigned de 32 bits. O resultado é `NULL` se o argumento é `NULL`. O argumento esperado é uma string e será tratado como tal se não for.

```
mysql> SELECT CRC32('MySQL');
        -> 3259397556
```

`CRC32()` está disponível a partir do MySQL 4.1.0.

`RAND()`
`RAND(N)` Retorna um valor de ponto flutuante aleatório na faixa de `0` a `1.0`. Se um argumento inteiro `N` é especificado, ele é usado como uma semente (produzindo uma sequência repetitiva):

```
mysql> SELECT RAND();
        -> 0.9233482386203
mysql> SELECT RAND(20);
        -> 0.15888261251047
mysql> SELECT RAND(20);
        -> 0.15888261251047
mysql> SELECT RAND();
        -> 0.63553050033332
mysql> SELECT RAND();
        -> 0.70100469486881
```

Você não pode usar uma coluna com valores `RAND()` em uma cláusula `ORDER BY`, pois `ORDER BY` avaliaria a coluna múltiplas vezes. Na versão 3.23 você pode fazer: `SELECT * FROM nome_tabela ORDER BY RAND()`

Isto é útil para obter um amostra aleatória de um conjunto `SELECT * FROM tabela1,tabela2 WHERE a=b AND c<d ORDER BY RAND() LIMIT 1000`.

Note que um `RAND()` em uma cláusula `WHERE` será reavliado toda vez que `WHERE` é executado.

`RAND()` não é um gerador de números aletatórios perfeito, mas é um modo rápido de se gerar números aleatórios ad hoc que serão portáveis entre plataformas para a mesma versão do MySQL.

`LEAST(X,Y,...)`

Com dois ou mais argumentos, retorna o menor (valor-mínimo) argumento. Os argumentos são comparados usando as seguintes regras:

- Se o valor de retorno é usado em um contexto `INTEGER`, ou todos argumentos são valores inteiro, eles são comparados como inteiros.
- Se o valor de retorno é usado em um contexto `REAL`, ou todos argumentos são valores reais, eles são comparados como inteiros.
- Se qualquer um dos argumento for uma string caso-sensitivo, os argumentos são comparados como strings caso-sensitivo.
- Nos outros casos, os argumentos são comparados como strings caso-insensitivo:

```
mysql> SELECT LEAST(2,0);
        -> 0
mysql> SELECT LEAST(34.0,3.0,5.0,767.0);
        -> 3.0
mysql> SELECT LEAST("B","A","C");
        -> "A"
```

Em versões do MySQL anteriores a versão 3.22.5, você pode usar `MIN()` no lugar de `LEAST`.

`GREATEST(X,Y,...)`

Retorna o maior (valor máximo) argumento. Os argumentos são comparados usando as mesmas regras do `LEAST`:

```
mysql> SELECT GREATEST(2,0);
        -> 2
mysql> SELECT GREATEST(34.0,3.0,5.0,767.0);
        -> 767.0
mysql> SELECT GREATEST("B","A","C");
        -> "C"
```

Em versões do MySQL anteriores a versão 3.22.5, você pode usar `MAX()` no lugar de `GRATEST`.

`DEGREES(X)`

Retorna o argumento `X`, convertido de radianos para graus:

```
mysql> SELECT DEGREES(PI());
        -> 180.000000
```

`RADIANS(X)`

Retorna o argumento `X`, convertido de graus para radianos:

```
mysql> SELECT RADIANS(90);
        -> 1.570796
```

`TRUNCATE(X,D)`

Retiorna o número `X`, truncado para `D` casas decimais. Se `D` é `0`, o resultado não terá ponto deciaml ou prate fracionária:

```
mysql> SELECT TRUNCATE(1.223,1);
        -> 1.2
mysql> SELECT TRUNCATE(1.999,1);
        -> 1.9
mysql> SELECT TRUNCATE(1.999,0);
        -> 1
mysql> SELECT TRUNCATE(-1.999,1);
        -> -1.9
```

A partir do MySQL 3.23.51 todos o números são arredondados para zero.

Se `D` é negativo, então D numeros da parte inteira são zerados:

```
mysql> SELECT TRUNCATE(122,-2);
       -> 100
```

Note que como os números decimais não são normalmente armazenados como números exatos, mas como valores de dupla precisão, você pode obter o seguinte resultado:

```
mysql> SELECT TRUNCATE(10.28*100,0);
       -> 1027
```

O resultado acima acontece porque 10.28 é, na verdade, armazenado como 10.2799999999999999.

6.3.4 Funções de Data e Hora

Esta seção descreve as funções que podem ser usadas para manipular valores temporais.

Veja Seção 6.2.2 [Tipos de data e hora], Página 489 para uma descrição da faixa de valores que cada tipo tem e os formatos válidos nos quais valores de data e hora podes ser especificados.

Aqui está um exemplo que usa funções de data. A consulta seguinte seleciona todos os registros com um valores em uma coluna `col_data` dentro dos últimos 30 dias:

```
mysql> SELECT algo FROM nome_tabela
           WHERE TO_DAYS(NOW()) - TO_DAYS(col_data) <= 30;
```

(Note que a consulta também selecionará registros com datas futuras.)

Funções que esperam valores de data normaemente aceitaram valores datetime e ignoram a parte da hora. Funções que esperam valores de hora normalmente aceitarão valores datetime e ignoram a parte da data.

Funções que retornam a data ou hora atual são avaliadas apenas uma vez por consulta, no inicio da sua execução. Isto significa que várias referências a uma função com `NOW()` dentro de uma mesma consulta sempre produzirá o memo resultado. Este princípio também se aplica a `CURDATE()`, `CURTIME()`, `UTC_DATE()`, `UTC_TIME()`, `UTC_TIMESTAMP()`, e qualquer um dos seus sinônimos.

A faixa do valor retornado na seguinte descrição da função se aplica a datas completas. Se uma data é um valor "zero" ou uma data incompleta tal como `'2001-11-00'`, funções que

extraem parte de uma data podem retornam 0. Por exemplo, `DAYOFMONTH('2001-11-00')` retorna 0.

`DATE(expr)`

Extrai a parte da data da expressão date ou datetime em `expr`.

```
mysql> SELECT DATE('2003-12-31 01:02:03');
        -> '2003-12-31'
```

`DATE()` está disponível a partir do MySQL 4.1.1.

`TIME(expr)`

Extrai a parte da hora da expressão time ou datetime em `expr`.

```
mysql> SELECT TIME('2003-12-31 01:02:03');
-> '01:02:03'
mysql> SELECT TIME('2003-12-31 01:02:03.000123');
        -> '01:02:03.000123'
```

`TIME()` está disponível a partir do MySQL 4.1.1.

`TIMESTAMP(expr)`
`TIMESTAMP(expr,expr2)`

Com um argumento, retorna a expressão date ou datetime em `expr` como um valor datetime. Com dois argumentos, adiciona a expressão time e `expr2` à expressão date ou datetime em `expr` e retorna um valor datetime.

```
mysql> SELECT TIMESTAMP('2003-12-31');
-> '2003-12-31 00:00:00'
mysql> SELECT TIMESTAMP('2003-12-31 12:00:00','12:00:00');
-> '2004-01-01 00:00:00'
```

`TIMESTAMP()` está disponível a partir do MySQL 4.1.1.

`DAYOFWEEK(data)`

Retorna o índice do dia da semana para `data` (`1` = Domingo, `2` = Segunda, ... `7` = Sábado). Estes valores de índices correspondem ao padrão ODBC.

```
mysql> SELECT DAYOFWEEK('1998-02-03');
        -> 3
```

`WEEKDAY(data)`

Retorna o índice do dia das semana para `data` (`0` = Segunda, `1` = Terça, ... `6` = Domingo):

```
mysql> SELECT WEEKDAY('1998-02-03 22:23:00');
        -> 1
mysql> SELECT WEEKDAY('1997-11-05');
        -> 2
```

`DAYOFMONTH(data)`

Retorna o dia do mês para `data`, na faixa de `1` até `31`:

```
mysql> SELECT DAYOFMONTH('1998-02-03');
        -> 3
```

`DAY(date)`

`DAY()` é um sinônimo para `DAYOFMONTH()`. Está disponível a partir do MySQL 4.1.1.

`DAYOFYEAR(data)`

Retorna o dia do ano para `data`, na faixa de `1` até `366`:

```
mysql> SELECT DAYOFYEAR('1998-02-03');
        -> 34
```

`MONTH(data)`

Retorna o mês para `data`, na faixa de `1` até `12`:

```
mysql> SELECT MONTH('1998-02-03');
        -> 2
```

`DAYNAME(data)`

Retorna o nome do dia da semana para `data`:

```
mysql> SELECT DAYNAME('1998-02-05');
        -> 'Thurday'
```

`MONTHNAME(data)`

Retorna o nome do mês para `data`:

```
mysql> SELECT MONTHNAME('1998-02-05');
        -> 'February'
```

`QUARTER(data)`

Retorna o trimaster para `data`, na faixa de `1` até `4`:

```
mysql> SELECT QUARTER('98-04-01');
        -> 2
```

`WEEK(data [,modo])`

A função retorna o número da semana para `date`. A forma de dois argumentos de `WEEK()` permite que você especifique se a semana começa no Domingo ou na Segunda e se o valor de retorno deve estar na faixa de 0-53 ou 1-5. Quando o argumento `modo` é omitido, o valor de uma variável de servidor `default_week_format` (ou 0 no MySQL 4.0 e mais novo) é assumido. Veja Seção 5.5.6 [SET OPTION], Página 460.

A seguinte tabela demonstra como o argumento `modo` funciona:

Valor	Significado
0	Semana começa no Domingo; retorna o valor na faixa de 0 a 53; semana 1 é a primeira semana neste ano.
1	Semana começa na Segunda; retorna o valor na faixa de 0 a 53; semana 1 é a primeira semana com mais de 3 dias neste ano
2	Semana começa no Domingo; retorna o valor na faixa de 1 a 53; semana 1 é a primeira semana neste ano.
3	Semana começa na Segunda; retorna o valor na faixa de 1 a 53; semana 1 é a primeira semana com mais de 3 dias neste ano.
4	Semana começa no Domingo; retorna o valor na faixa de 0 a 53; semana 1 é a primeira semana com mais de 3 dias neste ano.
5	Semana começa na Segunda; retorna o valor na faixa de 0 a 53; semana 1 é a primeira semana neste ano.
6	Semana começa no Domingo; retorna o valor na faixa de 0 a 53; semana 1 é a primeira semana que tenha mais de 3 dias neste ano.
7	Semana começa na Segunda; retorna o valor na faixa de 1 a 53; semana 1 é a primeira semana neste ano.

O valor `mode` de `3` pode ser usado a partir do MySQL 4.0.5. O valor `mode` de `4` e acima pode ser usado a partir do MySQL 4.0.17.

```
mysql> SELECT WEEK('1998-02-20');
        -> 7
mysql> SELECT WEEK('1998-02-20',0);
        -> 7
mysql> SELECT WEEK('1998-02-20',1);
        -> 8
mysql> SELECT WEEK('1998-12-31',1);
        -> 53
```

Nota: Na versão 4.0, `WEEK(#,0)` foi alterado para corresponder ao calendário americano. Antes `WEEK()` era calculada de forma errada para data no EUA. (Na verdade WEEK(#) e WEEK(#,0) era errado para todos os casos).

Note que se a data for a última semana do ano anterior, o MySQL retornará 0 se você não usar 2, 3, 6 ou 7 como argumento opcional `modo`:

```
mysql> SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0);
        -> 2000, 0
```

Pode-se questionar que o MySQL deveria retornar 52 para a função `WEEK()` ja que a data dada ocorre, na verdade, ma 52a. semana de 1999. Nós decidimos retornar 0 já que queremos que função retorne "o número da semana do ano dado". Isto faz com que o uso da função `WEEK()` seja seguro quando combinado com outras funções que extraiam um parte de uma data.

Se você prefere que o resultado seja avaliado em relacão ao ano que aontém o primeiro dia da semana de uma data dada, então você deve usar o 2, 3, 6 ou 7 como argumento opcional `modo`:

```
mysql> SELECT WEEK('2000-01-01',2);
        -> 52
```

Alternativamente você pode usar a função `YEARWEEK()`:

```
mysql> SELECT YEARWEEK('2000-01-01');
        -> 199952
mysql> SELECT MID(YEARWEEK('2000-01-01'),5,2);
        -> '52'
```

`WEEKOFYEAR(data)`

Retorna a semana da data como um número na faixa de `1` a `53`.

```
mysql> SELECT WEEKOFYEAR('1998-02-20');
        -> 8
```

`WEEKOFYEAR()` esta disponível a partir do MySQL 4.1.1.

`YEAR(data)`

Retorna o ano para `data` na faixa de `1000` a `9999`:

```
mysql> SELECT YEAR('98-02-03');
        -> 1998
```

`YEARWEEK(data)`
`YEARWEEK(data,inicio)`

Retorna o ano e a semana para a data. O argumento `inicio` funciona exatamente como o argumento `inicio` de `WEEK()`. Note que o ano pode ser diferente do ano no argumento data para a primeira e a última semana do ano:

```
mysql> SELECT YEARWEEK('1987-01-01');
        -> 198653
```

Note que o número da semana é diferente do que seria retornado pela função `WEEK()` (`0`) para os argumentos opcionais `0` ou `1`, já que `WEEK()` retorna a semana no centexto de um ano dado.

`HOUR(hora)`

Retorna a hora para `hora`. A faixa do valor retornado será de `0` a `23` para o valor hora do dia.

```
mysql> SELECT HOUR('10:05:03');
        -> 10
```

No entanto, a faixa dos valores `TIME` atualmente são muito grandes, assim `HOUR` pode retornar valores maior que `23`:

```
mysql> SELECT HOUR('272:59:59');
        -> 272
```

`MINUTE(hora)`

Retorna o minuto para `hora`, na faixa de `0` a `59`:

```
mysql> SELECT MINUTE('98-02-03 10:05:03');
        -> 5
```

`SECOND(hora)`

Retorna o segundo para `hora`, na faixa de `0` a `59`:

```
mysql> SELECT SECOND('10:05:03');
        -> 3
```

`MICROSECOND(expr)`

Retorna os microsegundos da expressão time ou datetime em `expr` como um número na faixa de `0` a `999999`.

```
mysql> SELECT MICROSECOND('12:00:00.123456');
        -> 123456
mysql> SELECT MICROSECOND('1997-12-31 23:59:59.000010');
        -> 10
```

`MICROSECOND()` está disponível a partir do MySQL 4.1.1.

`PERIOD_ADD(P,N)`

Adiciona `N` meses ao período `P` (no formato `AAMM` ou `AAAAMM`). Retorna um valor no formato `AAAAMM`.

Note que o argumento de período `P` **não** é um valor de data:

```
mysql> SELECT PERIOD_ADD(9801,2);
        -> 199803
```

`PERIOD_DIFF(P1,P2)`

Retorna o número de meses entre os períodos `P1` e `P2`. `P1` e `P2` devem estar no formato `AAMM` ou `AAAAMM`.

Note que os argumentos de período `P1` e `P2` **não** são valores de data:

```
mysql> SELECT PERIOD_DIFF(9802,199703);
        -> 11
```

`DATE_ADD(data,INTERVAL tipo expr)`
`DATE_SUB(data,INTERVAL tipo expr)`

Estas funções realizam operações aritméticas em datas.

A partir do MySQL 3.23, `INTERVAL expr tipo` é permitido nos dois lados do operador `+` se a expressao em ambos os lados é um valor date ou datetime. Para o operador `-`, `INTERVAL expr tipoe` é permitido apenas no lado direito, porque não faz sentido subtrair um valor date ou datetime de um intervalo. (Veja exemplo abaixo.)

`data` é um valor `DATETIME` ou `DATE` especificando a data de início. `expr` is an expressão especificando o intervala a ser adicionado ou subtraido da data de início. `expr` é uma string; ela pode iniciar com um '`-`' para intervalos negativos. `type` é uma palavra chave indicando como a expressão deve ser interpretada.

A seguinte tabela mostra como os argumentos `tipo` e `expr` se relacionam:

`tipo do` **valor**	**Formarto esperado da** `expr`
SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
MONTH	MONTHS
YEAR	YEARS
MINUTE_SECOND	'MINUTES:SECONDS'
HOUR_MINUTE	'HOURS:MINUTES'
DAY_HOUR	'DAYS HOURS'
YEAR_MONTH	'YEARS-MONTHS'
HOUR_SECOND	'HOURS:MINUTES:SECONDS'
DAY_MINUTE	'DAYS HOURS:MINUTES'
DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'
DAY_MICROSECOND	'DAYS.MICROSECONDS'
HOUR_MICROSECOND	'HOURS.MICROSECONDS'
MINUTE_MICROSECOND	'MINUTES.MICROSECONDS'
SECOND_MICROSECOND	'SECONDS.MICROSECONDS'
MICROSECOND	'MICROSECONDS'

Os valores do `tipo` `DAY_MICROSECOND`, `HOUR_MICROSECOND`, `MINUTE_MICROSECOND`, `SECOND_MICROSECOND` e `MICROSECOND` são permitidos após o MySQL 4.1.1.

O MySQL permite qualquer delimitador de pontuação no formato de `expr`. Os delimitadores mostrados na tabela são apenas sugeridos. Se o argumento `date` é um valor de `DATA` e seus cálculos envolvem apenas as partes `ANO`, `MÊS`, e `DIA`

(into é, nenhuma parte de hora), o resultado é um valor do tipo `DATE`. Senão, o resultado é um valor do tipo `DATETIME`:

```
mysql> SELECT '1997-12-31 23:59:59' + INTERVAL 1 SECOND;
        -> '1998-01-01 00:00:00'
mysql> SELECT INTERVAL 1 DAY + '1997-12-31';
-> '1998-01-01'
mysql> SELECT '1998-01-01' - INTERVAL 1 SECOND;
-> '1997-12-31 23:59:59'
mysql> SELECT DATE_ADD('1997-12-31 23:59:59',
        ->                 INTERVAL 1 SECOND);
        -> '1998-01-01 00:00:00'
mysql> SELECT DATE_ADD('1997-12-31 23:59:59',
        ->                 INTERVAL 1 DAY);
-> '1998-01-01 23:59:59'
mysql> SELECT DATE_ADD('1997-12-31 23:59:59',
        ->                 INTERVAL '1:1' MINUTE_SECOND);
        -> '1998-01-01 00:01:00'
mysql> SELECT DATE_SUB('1998-01-01 00:00:00',
->                 INTERVAL '1 1:1:1' DAY_SECOND);
-> '1997-12-30 22:58:59'
mysql> SELECT DATE_ADD('1998-01-01 00:00:00',
        ->                 INTERVAL '-1 10' DAY_HOUR);
        -> '1997-12-30 14:00:00'
mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
        -> '1997-12-02'
mysql> SELECT DATE_ADD('1992-12-31 23:59:59.000002',
        ->                 INTERVAL '1.999999' SECOND_MICROSECOND);
        -> '1993-01-01 00:00:01.000001'
```

Se você especificado um intervalo muito curto (não inclue todas as partes que seriam esperadas pelo intervalo para aquele `tipo`), MySQL assume que você não especificou a parte mais a esquerda do valor do intervalo. Por exemplo, se você especifica um `tipo DAY_SECOND`, o valor esperado de `expr` deverá ter as partes de dias, horas, minutos e segundos. Se você especifica um valor como `'1:10'`, MySQL assume que as partes do dia e da hora foram esquecidas e o valor representa minutos e segundos. Em outras palavras, `'1:10' DAY_SECOND` é interpretado de forma equivalente a `'1:10' MINUTE_SECOND`. Isto é análogo a forma que o MySQL interpreta valores `TIME` representado tempo decorrido no lugar de hora do dia.

Note que se você adicionar ou subtrair de uma data algo contendo uma parte de hora, o resultado é automaticamente convertido para um valor datetime:

```
mysql> SELECT DATE_ADD('1999-01-01', INTERVAL 1 DAY);
        -> '1999-01-02'
mysql> SELECT DATE_ADD('1999-01-01', INTERVAL 1 HOUR);
        -> '1999-01-01 01:00:00'
```

Se você utilizar datas mal formadas, o valor retornado `NULL`. Sê você adicionar `MONTH`, `YEAR_MONTH`, ou `YEAR` e a data resultante tiver um dia maior que o dia máximo para aquele mês, o dia é ajustado para o dia máximo no mês.

```
mysql> SELECT DATE_ADD('1998-01-30', interval 1 month);
        -> '1998-02-28'
```

Note pelo exemplo anterior que a palavra-chave `INTERVAL` e o especificador `tipo` não são caso sensitivo.

`ADDDATE(data,INTERVAL expr type)`
`ADDDATE(expr,dias)`

Quando chamada com a forma `INTERVAL` do segundo argumento, `ADDDATE()` é um sinônimo para `DATE_ADD()`. A função relcionada `SUBDATE()` é um sinônimo para `DATE_SUB()`.

```
mysql> SELECT DATE_ADD('1998-01-02', INTERVAL 31 DAY);
        -> '1998-02-02'
mysql> SELECT ADDDATE('1998-01-02', INTERVAL 31 DAY);
-> '1998-02-02'
```

A partir do MySQL 4.1.1, a segunda sintaxe é permitida, onde `expr` é uma expresão date ou datetime e `dias` é o número de dias a ser adicionado a `expr`.

```
mysql> SELECT ADDDATE('1998-01-02', 31);
-> '1998-02-02'
```

`ADDTIME(expr,expr2)`

`ADDTIME()` adiciona `expr2` a `expr` e retorna o resultado. `expr` é uma expressão date ou datetime, e `expr2` é uma expressão time.

```
mysql> SELECT ADDTIME("1997-12-31 23:59:59.999999", "1 1:1:1.000002");
        -> '1998-01-02 01:01:01.000001'
mysql> SELECT ADDTIME("01:00:00.999999", "02:00:00.999998");
        -> '03:00:01.999997'
```

`ADDTIME()` foi adicionado no MySQL 4.1.1.

`EXTRACT(tipo FROM data)`

A função `EXTRACT()` usa o mesmo tipo de intervalo especificado como `DATE_ADD()` ou `DATE_SUB()`, mas extrai partes da da data em vez de realizar aritimética de data.

```
mysql> SELECT EXTRACT(YEAR FROM "1999-07-02");
       -> 1999
mysql> SELECT EXTRACT(YEAR_MONTH FROM "1999-07-02 01:02:03");
       -> 199907
mysql> SELECT EXTRACT(DAY_MINUTE FROM "1999-07-02 01:02:03");
       -> 20102
mysql> SELECT EXTRACT(MICROSECOND FROM "2003-01-02 10:30:00.00123");
        -> 123
```

`DATEDIFF(expr,expr2)`
`TIMEDIFF(expr,expr2)`

`DATEDIFF()` retorna o número de dias entre a data inicial `expr` e a data final `expr2`. `expr` e `expr2` são expressões de datas ou data e hora. Apenas a parte da data dos valores sã usados no cálculo.

`TIMEDIFF()` retorna o tempo entre a hora inicial `expr` e a hora final `expr2`. `expr` e `expr2` são expressões de hora ou data e hora, mas ambas devem ser do mesmo tipo.

```
mysql> SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');
        -> 1
mysql> SELECT DATEDIFF('1997-11-31 23:59:59','1997-12-31');
        -> -30
mysql> SELECT TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001
        -> '-00:00:00.000001'
mysql> SELECT TIMEDIFF('1997-12-31 23:59:59.000001','1997-12-30 01:01:01.
        -> '46:58:57.999999'
```

`DATEDIFF()` e `TIMEDIFF()` foram adicionados no MySQL 4.1.1.

`TO_DAYS(data)`

Dada uma data `data`, retorna o número do dia (o número de dias desde o ano 0);

```
mysql> SELECT TO_DAYS(950501);
        -> 728779
mysql> SELECT TO_DAYS('1997-10-07');
        -> 729669
```

`TO_DAYS()` não pode ser usado com valores que orecedem o advento do calendario Gregoriano (1582), porque ele não leva em conta os dias perdidos quando o calendário foi mudado.

`FROM_DAYS(N)`

Dado um número de dia `N`, retorna um valor `DATE`:

```
mysql> SELECT FROM_DAYS(729669);
        -> '1997-10-07'
```

`FROM_DAYS()` não pode ser usado com valores que orecedem o advento do calendario Gregoriano (1582), porque ele não leva em conta os dias perdidos quando o calendário foi mudado.

`DATE_FORMAT(data,formato)`

Formata o valor de `data` de acordo com a string `formato` string. Os seguintes identificadores podem ser utilizados na string `formato`:

Specifier	Description
%M	Nome do mês (`January`..`December`)
%W	Nome da semana (`Sunday`..`Saturday`)
%D	Dia do mês com sufixo Inglês (`0th`, `1st`, `2nd`, `3rd`, etc.)
%Y	Ano, numerico, 4 digitos
%y	Ano, numerico, 2 digitos

%X	Ano para a semana onde o Domingo é o primeiro dia da semana, numerico, 4 digitos; usado com %V
%x	Ano para a semana onde a segunda é o primeiro dia da semana, numerico, 4 digitos; usado com %v
%a	Nome da semana abreviado (Sun..Sat)
%d	Dia do mês, numerico (00..31)
%e	Dia do mês, numerico (0..31)
%m	Mês, numerico (00..12)
%c	Mês, numerico (0..12)
%b	Nome do mês abreviado (Jan..Dec)
%j	Dia do ano (001..366)
%H	Hora (00..23)
%k	Hora (0..23)
%h	Hora (01..12)
%I	Hora (01..12)
%l	Hora (1..12)
%i	Minutos, numerico (00..59)
%r	Tempo, 12-horas (hh:mm:ss seguido por AM ou PM)
%T	Tempo, 24-horas (hh:mm:ss)
%S	Segundos (00..59)
%s	Segundos (00..59)
%f	Microsegundos (000000..999999)
%p	AM ou PM
%w	Dia da semana (0=Domingo..6=Sabado)
%U	Semana(00..53), onde o Domingo é o primeiro dia da semana.
%u	Semana(00..53), onde a Segunda é o primeiro dia da semana.
%V	Semana(01..53), onde o Domingo é o primeiro dia da semana; usado com %X
%v	Semana(01..53), onde a Segunda é o primeiro dia da semana; usado com %x
%%	Um literal '%'.

Todos os outros caracteres são apenas copiados para o resultado, sem interpretação.

O especificador dr formato %f está disponível a partir do MySQL 4.1.1.

Como na versão 3.23 do MySQL, o caracter '%' é exigido antes dos caracteres de especificação de formato. Em versões anteriores do MySQL '%' era opcional.

A razão para a faixa de valores do mês e do dia começarem com zero é que o MySQL permite datas incompletas tais como '2004-00-00' serem armazenadas no MySQL 3.23.

```
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%W %M %Y');
        -> 'Saturday October 1997'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');
        -> '22:23:00'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%D %y %a %d %m %b %j');
```

```
        -> '4th 97 Sat 04 10 Oct 277'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%H %k %I %r %T %S %w');
        -> '22 22 10 10:23:00 PM 22:23:00 00 6'
mysql> SELECT DATE_FORMAT('1999-01-01', '%X %V');
        -> '1998 52'
```

`STR_TO_DATE(str,format)`

Esta é a função reversa da função `DATE_FORMAT()`. Ela pega uma string `str`, e um formato `format`, e retorna uma valor DATETIME. Os valores date, time, ou datetime contidos em `str` devem ser dados no formato indicado por `format`. Para o especificadores que podem ser usados em `format`, veja a tabela na descrição da função `DATE_FORMAT()`. Todos os outros caracteres serão apenas exibidos, não sendo interpretados. Se `str` contém um valor date, time, ou datetime ilegal, `STR_TO_DATE()` retorna `NULL`.

```
mysql> SELECT STR_TO_DATE('03.10.2003 09.20', '%d.%m.%Y %H.%i')
        -> 2003-10-03 09:20:00
mysql> SELECT STR_TO_DATE('10rap', '%crap')
        -> 0000-10-00 00:00:00
mysql> SELECT STR_TO_DATE('2003-15-10 00:00:00', '%Y-%m-%d %H:%i:%s')
        -> NULL
```

`STR_TO_DATE()` está disponível a partir do MySQL 4.1.1.

`GET_FORMAT(DATE | TIME | TIMESTAMP, 'EUR' | 'USA' | 'JIS' | 'ISO' | 'INTERNAL')`

Retorna uma string de formato. Esta função é útil combinado com as funções `DATE_FORMAT()` e `STR_TO_DATE()`, e quando configurarmos as variáveis do servidor `DATE_FORMAT`, `TIME_FORMAT` e `DATETIME_FORMAT`. Os três valores possíveis para o primeiro argumento e os cinco valores possíveis para o segundo argumento resultam em 15 strings de formato possíveis (para o especificador usado, veja a tabela na descrição da função `DATE_FORMAT()`):

Chamada da Função	**Resultado**
`GET_FORMAT(DATE,'USA')`	`'%m.%d.%Y'`
`GET_FORMAT(DATE,'JIS')`	`'%Y-%m-%d'`
`GET_FORMAT(DATE,'ISO')`	`'%Y-%m-%d'`
`GET_FORMAT(DATE,'EUR')`	`'%d.%m.%Y'`
`GET_FORMAT(DATE,'INTERNAL')`	`'%Y%m%d'`
`GET_FORMAT(TIMESTAMP,'USA')`	`'%Y-%m-%d-%H.%i.%s'`
`GET_FORMAT(TIMESTAMP,'JIS')`	`'%Y-%m-%d %H:%i:%s'`
`GET_FORMAT(TIMESTAMP,'ISO')`	`'%Y-%m-%d %H:%i:%s'`
`GET_FORMAT(TIMESTAMP,'EUR')`	`'%Y-%m-%d-%H.%i.%s'`
`GET_FORMAT(TIMESTAMP,'INTERNAL')`	`'%Y%m%d%H%i%s'`
`GET_FORMAT(TIME,'USA')`	`'%h:%i:%s %p'`
`GET_FORMAT(TIME,'JIS')`	`'%H:%i:%s'`
`GET_FORMAT(TIME,'ISO')`	`'%H:%i:%s'`
`GET_FORMAT(TIME,'EUR')`	`'%H.%i.%S'`
`GET_FORMAT(TIME,'INTERNAL')`	`'%H%i%s'`

Formato ISO é do ISO ISO 9075, não do ISO 8601.

```
mysql> SELECT DATE_FORMAT('2003-10-03', GET_FORMAT(DATE, 'EUR')
        -> '03.10.2003'
mysql> SELECT STR_TO_DATE('10.31.2003', GET_FORMAT(DATE, 'USA'))
        -> 2003-10-31
mysql> SET DATE_FORMAT=GET_FORMAT(DATE, 'USA'); SELECT '2003-10-31';
        -> 10-31-2003
```

`GET_FORMAT()` está disponível a partir do MySQL 4.1.1. Veja Veja Seção 5.5.6 [SET OPTION], Página 460.

`SUBDATE(date,INTERVAL expr type)`
`SUBDATE(expr,days)`

Quando chamado com a forma `INTERVAL` do segunto argumento, `SUBDATE()` é um sinonimo para `DATE_SUB()`.

```
mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
    -> '1997-12-02'
mysql> SELECT SUBDATE('1998-01-02', INTERVAL 31 DAY);
    -> '1997-12-02'
```

A partir do MySQL 4.1.1, a segunda sintaxe é permitida, onde `expr` é uma expressão date ou datetime e `days` é o número de dias a ser subtraído de `expr`.

```
mysql> SELECT SUBDATE('1998-01-02 12:00:00', 31);
-> '1997-12-02 12:00:00'
```

`SUBTIME(expr,expr2)`

`SUBTIME()` subtrai `expr2` de `expr` e retorna o resultado. `expr` é uma expressão date ou datetime, e `expr2` é uma expressão time.

```
mysql> SELECT SUBTIME("1997-12-31 23:59:59.999999", "1 1:1:1.000002");
-> '1997-12-30 22:58:58.999997'
mysql> SELECT SUBTIME("01:00:00.999999", "02:00:00.999998");
-> '-00:59:59.999999'
```

`SUBTIME()` foi adicionado no MySQL 4.1.1.

`TIME_FORMAT(hora,formato)`

É usado como a função `DATE_FORMAT()` acima, mas a string de `formato` pode conter apenas os especificadores de formato que tratam de horas, minutos e segundos. Outros especificadores produzem um valor `NULL` ou `0`.

Se o valor `time` contém uma hora que é maior que `23`, os especificadores de formato de hora `%H` e `%k` produzem um valor maior que a faixa como de `0..23`. O outro especificador do formato de hora produz o valor da hora módulo 12:

```
mysql> SELECT TIME_FORMAT('100:00:00', '%H %k %h %I %l');
        -> '100 100 04 04 4'
```

`LAST_DAY(data)`

Pega um valor date ou datetime e retorna o valor correspondente para o último dia do mês. Retorna `NULL` se o argumento é invalido.

```
mysql> SELECT LAST_DAY('2003-02-05'), LAST_DAY('2004-02-05');
        -> '2003-02-28', '2004-02-29'
mysql> SELECT LAST_DAY('2004-01-01 01:01:01');
```

```
        -> '2004-01-31'
mysql> SELECT LAST_DAY('2003-03-32');
        -> NULL
```

`LAST_DAY()` está disponível a partir do MySQL 4.1.1.

`MAKEDATE(ano,diadoano)`
: Retorna uma data, dado os valores da ano e dia do ano. `diadoano` deve ser maior que 0 ou o resultado será `NULL`.

```
mysql> SELECT MAKEDATE(2001,31), MAKEDATE(2001,32);
        -> '2001-01-31', '2001-02-01'
mysql> SELECT MAKEDATE(2001,365), MAKEDATE(2004,365);
-> '2001-12-31', '2004-12-30'
mysql> SELECT MAKEDATE(2001,0);
        -> NULL
```

`MAKEDATE()` está disponível a partir do MySQL 4.1.1.

`MAKETIME(hora,minuto,segundo)`
: Retorna um valor time calculado a partir dos argmentos `hora`, `minuto` e `segundo`.

```
mysql> SELECT MAKETIME(12,15,30);
-> '12:15:30'
```

`MAKETIME()` está disponível a partir do MySQL 4.1.1.

`CURDATE()`
`CURRENT_DATE`
`CURRENT_DATE()`
: Retorna a data atual como um valor no formato `'YYYY-MM-DD'` ou `YYYYMMDD`, dependendo se a função é usada num contexto numérico ou de string.

```
mysql> SELECT CURDATE();
        -> '1997-12-15'
mysql> SELECT CURDATE() + 0;
        -> 19971215
```

`CURTIME()`
`CURRENT_TIME`
`CURRENT_TIME()`
: Retorna a hora atual como um valor no formato `'HH:MM:SS'` ou `HHMMSS`, dependo se a função é usada em um contexto numérico ou como string:

```
mysql> SELECT CURTIME();
        -> '23:50:26'
mysql> SELECT CURTIME() + 0;
        -> 235026
```

NOW()
SYSDATE()
CURRENT_TIMESTAMP
CURRENT_TIMESTAMP()
LOCALTIME
LOCALTIME()
LOCALTIMESTAMP
LOCALTIMESTAMP()
: Retorna a data e hora atual como um valor no formato 'YYYY-MM-DD HH:MM:SS' ou YYYYMMDDHHMMSS, dependendo se a função é utilizada num contexto numérico ou de string.

```
mysql> SELECT NOW();
        -> '1997-12-15 23:50:26'
mysql> SELECT NOW() + 0;
        -> 19971215235026
```

UNIX_TIMESTAMP()
UNIX_TIMESTAMP(data)
: Se chamado sem argumento, retorna um tipo timestamp do Unix (segundos desde '1970-01-01 00:00:00' GMT) como um inteiro sem sinal. Se UNIX_TIMESTAMP() é chamada com um argumento data, é retornado o valor do argumento como segundo desde '1970-01-01 00:00:00' GMT. data pode ser um string DATE, uma string DATETIME, um TIMESTAMP, ou um número no formato YYMMDD ou YYYYMMDD na hora local:

```
mysql> SELECT UNIX_TIMESTAMP();
        -> 882226357
mysql> SELECT UNIX_TIMESTAMP('1997-10-04 22:23:00');
        -> 875996580
```

Qaundo UNIX_TIMESTAMP é usado em uma coluna TIMESTAMP, a função retorna o valor timestamp interno diretamente, sem nenhuma conversão "string-para-unix-timestamp" implicita. Se você passar uma data fora da faixa para UNIX_TIMESTAMP(), a função irá retornar 0, mas por favor note que só verificações básicas são realizadas. (ano 1970-2037, mês 01-12, dia 01-31).

Se você subtrair colunas UNIX_TIMESTAMP(), você pode querer mudar o resultado para inteiro com sinal. Veja Seção 6.3.5 [Funções de tipagem], Página 544.

FROM_UNIXTIME(unix_timestamp)
FROM_UNIXTIME(unix_timestamp,format)
: Retorna a representação do argumento unix_timestamp como um valor no formato 'YYYY-MM-DD HH:MM:SS' ou YYYYMMDDHHMMSS, dependendo de do contexto em que a funçõ é utilizada:

```
mysql> SELECT FROM_UNIXTIME(875996580);
        -> '1997-10-04 22:23:00'
mysql> SELECT FROM_UNIXTIME(875996580) + 0;
        -> 19971004222300
```

Se o `formato` é dado o resultado é formatado de acordo com a string `formato`. `formato` pode conter os especificadores listados acima para a função `DATE_FORMAT()`

```
mysql> SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(),
    ->                      '%Y %D %M %h:%i:%s %x');
    -> '2003 6th August 06:22:58 2003'
```

`SEC_TO_TIME(seconds)`

Retorna o argumento `segundos`, convertido em horas, minutos e segundos como um valor no formato `'HH:MM:SS'` ou `HHMMSS`, dependendo do contexto em que a função é utilizada:

```
mysql> SELECT SEC_TO_TIME(2378);
        -> '00:39:38'
mysql> SELECT SEC_TO_TIME(2378) + 0;
        -> 3938
```

`TIME_TO_SEC(time)`

Retorna o argumento `time`, convertido em segundos:

```
mysql> SELECT TIME_TO_SEC('22:23:00');
        -> 80580
mysql> SELECT TIME_TO_SEC('00:39:38');
        -> 2378
```

`UTC_DATE`
`UTC_DATE()`

Retorna a data UTC atual como um valor no formato `'YYYY-MM-DD'` ou `YYYYMMDD`, dependendo se a função é usada emum contexto string ou numérico:

```
mysql> SELECT UTC_DATE(), UTC_DATE() + 0;
-> '2003-08-14', 20030814
```

`UTC_DATE()` está disponível a partir do MySQL 4.1.1.

`UTC_TIME`
`UTC_TIME()`

Retorna a hora UTC atual como um valor no formato `'HH:MM:SS'` ou `HHMMSS`, dependendo se a função é usada em um contexto string ou numérico:

```
mysql> SELECT UTC_TIME(), UTC_TIME() + 0;
-> '18:07:53', 180753
```

`UTC_TIME()` está disponível a partir do MySQL 4.1.1.

`UTC_TIMESTAMP`
`UTC_TIMESTAMP()`

Retorna a data e hora UTC atual como um valor no formato `'YYYY-MM-DD HH:MM:SS'` ou `YYYYMMDDHHMMSS`, dependendo se a função é usada em um contexto string ou numérico:

```
mysql> SELECT UTC_TIMESTAMP(), UTC_TIMESTAMP() + 0;
-> '2003-08-14 18:08:04', 20030814180804
```

`UTC_TIMESTAMP()` está disponível a partir do MySQL 4.1.1.

6.3.5 Funções de Conversão

As funções `CAST()` e `CONVERT()` devem ser usada para tomar um valor de um tipo e produzir um valor de outro tipo. As suas sintaxes são as seguintes:

```
CAST(expressão AS tipo)
CONVERT(expressão,tipo)
CONVERT(expr USING transcoding_name)
```

O valor `tipo` pode ser um dos seguintes:

- `BINARY`
- `CHAR`
- `DATE`
- `DATETIME`
- `SIGNED {INTEGER}`
- `TIME`
- `UNSIGNED {INTEGER}`

`CAST()` e `CONVERT()` estão disponíveis a partir do MySQL 4.0.2. O tipo de conversão `CHAR` está disponível a partir do versão 4.0.6. A forma `USING` de `CONVERT()` está disponível a partir da versão 4.1.0.

`CAST()` e `CONVERT(... USING ...)` são da sintaxe SQL-99. A forma não-`USING` de `CONVERT()` é da sintaxe ODBC.

`CAST()` é da sintaxe SQL-99 syntax e `CONVERT()` é da sintaxe ODBC.

As funções de conversão são principalmente úteis quando você deseja criar uma coluna com um tipo específico em uma `CREATE ... SELECT`:

```
CREATE TABLE nova_tabela SELECT CAST('2000-01-01' AS DATE);
```

As funções também podem ser úteis para ordenar colunas `ENUM` na ordem lexicográfica. Normalmente a ordenação das colunas `ENUM` ocorrem usando os valores numéricos internos. Converter os valores para `CHAR` resultam em uma ordenação lexicográfica:

```
SELECT enum_col FROM tbl_name ORDER BY CAST(enum_col AS CHAR);
```

`CAST(string AS BINARY)` é a mesma coisa que `BINARY string`. `CAST(expr AS CHAR)` trata a expressão como uma string com o conjunto de caracteres padrão.

NOTA: No MysQL 4.0 o `CAST()` para `DATE`, `DATETIME` ou `TIME` só marca a coluna para ser um tipo específico mas não altera o valor da coluna.

No MySQL 4.1.0 o valor será convertido para a coluna correta quando for enviado para o usuário (este é um recurso de como o novo protocolo na versão 4.1 envia as informações de data para o cliente):

```
mysql> SELECT CAST(NOW() AS DATE);
       -> 2003-05-26
```

Em versões futuras do MySQL (provavelmente 4.1.2 ou 5.0) iremos corrigir o fato de que `CAST` também altera o resultado se você usá-lo como parte de uma expressão mais complexa, como `CONCAT("Data: ",CAST(NOW() AS DATE))`.

Você não deve utilizar `CAST()` para extrair dados em formatos diferentes, mas sim para usar funções strins como `LEFT` ou `EXTRACT()`. Veja Seção 6.3.4 [Funções de data e tempo], Página 529.

Para converter uma string para um valor numérico, normalmente não é necessário se fazer nada; apenas use a string como se fosse um número:

```
mysql> SELECT 1+'1';
       -> 2
```

Se você usar um número em um contexto string, o número será convertido automaticamente para uma string `BINARY`.

```
mysql> SELECT CONCAT("hello you ",2);
       ->  "hello you 2"
```

O MySQL suporta aritimético com valores de 64 bits com sinal e sem sinal. Se você está usando operações numéricas (como `+`) e um dos operandos é `unsigned integer` (inteiro sem sinal), o resultado também será sem sinal (unsigned). Você pode forçar o tipo usando os operadores de conversão `SIGNED` e `UNSIGNED` para converter a operação para um inteiro de 64 bits com sinal e sem sinal, respectivamente.

```
mysql> SELECT CAST(1-2 AS UNSIGNED)
        -> 18446744073709551615
mysql> SELECT CAST(CAST(1-2 AS UNSIGNED) AS SIGNED);
        -> -1
```

Note que se um dos operandos for um valor de ponto flutuante o resultado é um valor de ponto flutuante e não é afetado pela regra acima. (Neste contexto `DECIMAL()` é considerado um valor de ponto flutuante).

```
mysql> SELECT CAST(1 AS UNSIGNED) -2.0;
        -> -1.0
```

Se você estiver utilizando uma string em uma operação aritimética, ela é convertida para um número de ponto flutuante.

O tratamento de valores sem sinais foi mudado no MySQL 4.0 para suportar valores `BIGINT` apropriadamente. Se você tiver algum código que deseja executar no MySQL 4.0 e 3.23 (casos em que você provavelmente não poderá usar a função `CAST()`), você pode utilizar o seguinte truque para conseguir um resultado com sinal quando subtraindo duas colunas do tipo unsigned integer (inteiro sem sinal):

```
SELECT (coluna_sem_sinal_1+0.0)-(coluna_sem_sinal_2+0.0);
```

A idéia é que as colunas sejam convertidas para valores de ponto flutuante antes da subtração ocorrer.

Se você tiver algum problema com colunas `UNSIGNED` no seu aplicação MySQL antiga ao portar para o MySQL 4.0, você pode usar a opção `--sql-mode=NO_UNSIGNED_SUBTRACTION` ao iniciar `mysqld`. Note, no entanto, que enquanto você utilizar esta opção, não será possível conseguir um uso efetivo do tipo de coluna `BIGINT UNSIGNED`.

`CONVERT()` com `USING` é usado para converter dados entre diferentes conjuntos de caracteres. No MySQL, nomes trancodificados são o mesmo que o nome do conjunto de caracteres correspondentes. Por exemplo, esta instrução converte a string `'abc'` no conjunto de caracteres padrão do servidor na string correspondente no conjunto de caracteres `utf8`:

```
SELECT CONVERT('abc' USING utf8);
```

6.3.6 Outras Funções

6.3.6.1 Funções Binárias

O MySQL utiliza aritimética `BIGINT` (64bits) para operações binárias, assim estes operadores possuem uma faixa máxima de 64 bits.

`|` Operador binário OR

```
mysql> SELECT 29 | 15;
        -> 31
```

O resultado é um inteiro sem sinal de 64 bits.

`&` Operado binário AND

```
mysql> SELECT 29 & 15;
        -> 13
```

O resultado é um inteiro sem sinal de 64 bits.

`^` Operado binário XOR

```
mysql> SELECT 1 ^ 1;
        -> 0
mysql> SELECT 1 ^ 0;
        -> 1
mysql> SELECT 11 ^ 3;
        -> 8
```

O resultado é um inteiro sem sinal de 64 bits.

`XOR` foi adicionado na versão 4.0.2.

`<<`

```
Desloca um número BIGINT (muito grande) a esquerda:
mysql> SELECT 1 << 2;
        -> 4
```

O resultado é um inteiro sem sinal de 64 bits.

`>>` Desloca um número `BIGINT` (muito grande) a direita:

```
mysql> SELECT 4 >> 2;
        -> 1
```

O resultado é um inteiro sem sinal de 64 bits.

`~` Inverte todos os bits:

```
mysql> SELECT 5 & ~1;
        -> 4
```

O resultado é um inteiro sem sinal de 64 bits.

`BIT_COUNT(N)`

Retorna o número de bits que são passados no argumento `N`:

```
mysql> SELECT BIT_COUNT(29);
        -> 4
```

6.3.6.2 Funções Diversas

`DATABASE()`

Retorna o nome do banco de dados atual:

```
mysql> SELECT DATABASE();
        -> 'test'
```

Se nenhum banco de dados estiver selecionado, `DATABASE()` retorna `NULL` a partir do MySQL 4.1.1, e uma string vazia em versões anteriores.

`USER()`
`SYSTEM_USER()`
`SESSION_USER()`

Retorna o nome do usuário MySQL e nome de máquina atual:

```
mysql> SELECT USER();
        -> 'davida@localhost'
```

O valor indica o nome do usuário que você especificou ao conectar ao servidor e a máquina cliente da qual você se conectou. (Antes do MySQL versão 3.22.11, o valor da função não inclui o nome da máquina cliente.)

Você pode extrair apenas a parte do nome do usuário, desconsiderando se o valor inclui a parte do nome de máquina, desta forma:

```
mysql> SELECT SUBSTRING_INDEX(USER(),"@",1);
        -> 'davida'
```

`CURRENT_USER()`

Retorna o nome do usuário e o nome de máquina com os quais a sessão atual foi autenticada. Este valor corresponde a conta que é usada para acessar seu privilégio de acessos. Ela pode ser diferente do valor de `USER()`.

```
mysql> SELECT USER();
        -> 'davida@localhost'
mysql> SELECT * FROM mysql.user;
        -> ERROR 1044: Access denied for user: '@localhost' to database '
mysql> SELECT CURRENT_USER();
        -> '@localhost'
```

O exemplo ilustra que embora o cliente tenha especificado um nome de usuário `davida` (como indicado pelo valor da função `USER()`), o servidor autenticou o cliente usando uma conta de usuário anônimo (como visto pela parte vazia no nome de usuário do valor `CURRENT_USER()`). Um modos de isto ocorrer é que não haja uma conta listada na tabela de permissões para `davida`.

`PASSWORD(str)`
`OLD_PASSWORD(str)`

Calcula a senha a partir de senha `str` em texto puro. Está é a função que é utilizada para criptografar a senha do MySQL para armazenamento na coluna `Password` da tabela de permissões `user`

```
mysql> SELECT PASSWORD('badpwd');
        -> '7f84554057dd964b'
```

A criptografia de `PASSWORD()` não e reversível.

`PASSWORD()` não realiza a criptografia da senha da mesa maneira que as senhas Unix são criptografadas. Veja `ENCRYPT()`.

Note: A função `PASSWORD()` é usada pelo sistema de autentificação no servidor MySQL, você `NÃO` deve uitlizá-las em suas próprias aplicações. Para este propósito utilize `MD5()` ou `SHA1()`. Veja também `RFC-2195` para maiores informações sobre o tratamento de senha e autenticação segura em suas aplicações.

`ENCRYPT(str[,salt])`

Criptografa `str` utilizando a chamada de sistema `crypt()` do Unix. O argumento `salt` deve ser uma string com dois caracteres. (Na versão 3.22.16 do MySQL, `salt` deve ser maior que dois caracteres.)

```
mysql> SELECT ENCRYPT("hello");
        -> 'VxuFAJXVARROc'
```

`ENCRYPT()` ignora tudo depois dos primeiros 8 caracteres de `str`, pelo menos em alguns sistemas. Este comportamento é determinado pela implementação da chamada de sistema `crypt()`.

Se `crypt()` não estiver disponível no seu sistema, `ENCRYPT()` sempre retorna `NULL`. Devido a isto recomendamos que você use `MD5()` ou `SHA1()` em vez dos existentes em sua plataforma.

`ENCODE(str,senha_str)`

Criptografa `str` usando `senha_str` como a senha. Para descriptografar o resultado, utilize `DECODE()`.

O resultado é uma string binária do mesmo tamanho de `str`. Se você deseja salvá-la em uma coluna, use uma coluna do tipo `BLOB`.

`DECODE(cript_str,senha_str)`

Descriptografa o string criptografada `cript_str` usando `senha_str` como a senha. `cript_str` deve ser uma string retornada de `ENCODE()`.

`MD5(string)`

Calcula um checksum MD5 de 128 bits para a string. O valor é retornado como um número hexadecimal de 32 digitos que pode, por exemplo, ser usado como uma chave hash:

```
mysql> SELECT MD5("testing");
        -> 'ae2b1fca515949e5d54fb22b8ed95575'
```

Este é o "RSA Data Security, Inc. MD5 Message-Digest Algorithm".

`SHA1(string)`
`SHA(string)`

Calcula um checksum SHA1 de 160 bit para a string, como descrito no RFC 3174 (Algoritmo Hash de Segurança). O valor é retornado como um número hexadecial de 40 digitos, or `NULL` no caso do argumento ser `NULL` . Uma das possibilidades para o uso desta função é a chave hash. Você também pode usá-lo como uma função segura de criptografia para armazenar senhas.

```
mysql> SELECT SHA1("abc");
        -> 'a9993e364706816aba3e25717850c26c9cd0d89d'
```

`SHA1()` foi adicionado na versão 4.0.2, e pode ser considerada um equivalente ao `MD5()` com criptografia mais segura. `SHA()` é um sinônimo para `SHA1()`.

`AES_ENCRYPT(string,string_chave)`
`AES_DECRYPT(string,string_chave)`

Estas funções permitem criptografia/descriptografia de dados usando o algoritmo oficial AES (Padrão Avançado de Criptografia), antes conhecido como Rijndael. Criptgrafia com uma chave de 128 bits podem ser usadas, mas você pode extendê-la para 256 bits através da fonte. Nós escolhemos 128 bits porque é muito mais rápido e é bastante seguro.

Os argumentos de entrada podem ser de qualquer tamanho. Se ambos argumentos são `NULL`, o resultado desta função tam bém será `NULL`.

Como o AES é um algorítimo de nível de bloco, padding é usado para codificar strings de tamanho ímpares e então a string resultante pode ser calculada como 16*(trunc(tamanho_string/16)+1).

Se `AES_DECRYPT()` detectar dados inválidos ou padding incorreto, ela retorna `NULL`. No entanto, é possível para o `AES_DECRYPT()` retornar um valor não-`NULL` (possivelmente lixo) se os dados de entrada ou a chave eram inválidos

Você pode usar as funções AES para armazenar dados de forma criptografada modificando as suas consultas:

```
INSERT INTO t VALUES (1,AES_ENCRYPT('text','password'));
```

Você pode obter mais segurança não transferindo a chave em suas conexões a cada consulta, o que pode ser conseguido armazenando-o em varáveis do lado do servidor na hora das conexão.

```
SELECT @password:='my password';
INSERT INTO t VALUES (1,AES_ENCRYPT('text',@password));
```

`AES_ENCRYPT()` e `AES_DECRYPT()` foram adicionados na versão 4.0.2, e podem ser considerados a função de criptografia mais segura atualmente disponível no MySQL.

`DES_ENCRYPT(string_para_ciptografar [, (numero_chave | chave_string) ] )`

Criptografa a string com a chave dada utilizando o algortimo Triplo-DES.

Note que esta função só funciona se o MySQL tiver sido configurado com suporte a SSL. Veja Seção 4.4.10 [Conexões seguras], Página 268.

A chave de criptografia utilizada é escolhida da seguinte forma:

Argumento	**Descrição**
Somente um argumento	A primeira chave de `des-key-file` é utilizada.
Número da chave	A chave dada (0-9) de `des-key-file` é utilizada.
string	A `chave_string` dada será utilizada para criptografar `string_para_criptografar`.

O string retornada será uma string binária onde o primeiro caracter será `CHAR(128 | número_chave)`.

O 128 é adicionado para facilitar o reconhecimento da chave de criptografia. Se você usar uma chave string, `numéro_chave` será 127.

Havendo erro, esta função retorna `NULL`.

O tamanho da string para o resultado será `novo_tamanho= tamanho_orig + (8-(tamanho_orig % 8))+1`.

O `des-key-file` terá o seguinte formato:

```
numero_chave chave_string_des
numero_chave chave_string_des
```

Cada `numero_chave` deve ser um núero na faixa de 0 a 9. As linhas do arquivo podem estar em qualquer ordem. `chave_string_des` é a string que será usada para criptografar a mensagem. Entre o número e a chave deve haver pelo menos um espaço. A primeira chave é a chave padrão que será utilizada se não for especificada nenhuma chave como argumento para `DES_ENCRYPT()`

Você pode dizer ao MySQL para ler novos valores de arquivos de chave com o comando `FLUSH DES_KEY_FILE`. Isto exige o privilégio `Reload_priv`.

Um benefício de ter um conjunto de chaves padrões é que ele dá a aplicação um modo de verificar a existência de valores criptografados em colunas, sem dar ao usuário final o direito de descriptografar estes valores.

```
mysql> SELECT endereco_clientes FROM tabela_clientes WHERE
       cartao_credito_criptografado = DES_ENCRYPT("numero_cartao_credito"
```

`DES_DECRYPT(string_para_descriptografar [, chave_string])`

Derscritogra uma string criptografada com `DES_ENCRYPT()`.

Note que esta função só funciona se o MySQL tiver sido configurado com suporte SSL. Veja Seção 4.4.10 [Conexões seguras], Página 268.

Se nenhum argumento `chave_string` for dado, `DES_DECRYPT()` examina o primeiro byte da string criptografada para determinar o número de chave DES que foi usado para criptografar a string original, e então lê a chave de `des-key-file` para descriptografar a mensagem. Para isto funcionar o usuário deve ter o privilégio `SUPER`.

Se você passar para esta função um argumento `chave_string`, aquela string é usada como a chave para descriptografar a mensagem.

Se a `string_para_descriptografar` não se paracer com uma string criptografada, o MySQL retornará a `string_para_descriptografar` dada.

Havendo erro, esta função retorna `NULL`.

`COMPRESS(string_para_compactar)`

Compacta uma string

```
mysql> SELECT LENGTH(COMPRESS(REPEAT("a",1000)));
-> 21
1 row in set (0.00 sec)

mysql> SELECT LENGTH(COMPRESS(""));
-> 0
1 row in set (0.00 sec)

mysql> SELECT LENGTH(COMPRESS("a"));
```

```
-> 13
1 row in set (0.00 sec)

mysql> SELECT LENGTH(COMPRESS(REPEAT("a",16)));
-> 15
1 row in set (0.00 sec)
```

`COMPRESS()` foi adicionado no MySQL 4.1.1. Se exigido, o MySQL tem que ser compilado com uma biblioteca de compactação como `zlib`. Senão , o valor de retorno é sempre `NULL`.

O conteúdo da string compactada é armazenada da seguinte forma:

- Strings vazias são armazenadas como strings vazias
- Strings que não estão vazias são armazenadas como um string descompacatada de 4 byte de tamanho (low-byte-first) seguida pela string compactada com gzip. Se a string finaliza com espaço, adicionamos um '.' extra para evitar problemas com o corte do espaço final o resultado deve ser armazenado em um campo `CHAR` ou `VARCHAR`. O uso de `CHAR` ou `VARCHAR` para armazenar strings compactadas não é recomendado. É melhor usar uma coluna `BLOB`.

`UNCOMPRESS(string_para_descompactar)`

Descompacta uma string compactado pela função `COMPRESS()`

```
mysql> select UNCOMPRESS(COMPRESS("any string"));
-> 'any string'
1 row in set (0.00 sec)
```

`UNCOMPRESS()` foi adicionado no MySQL 4.1.1 Se exigido, o MySQL tem que ser compilado com uma biblioteca de compactação como `zlib`. Senão , o valor de retorno é sempre `NULL`.

`UNCOMPRESSED_LENGTH(string_compactada)`

Retorna o tamanho da string compactada antes da compactação

```
mysql> select UNCOMPRESSED_LENGTH(COMPRESS(REPEAT("a",30)));
-> 30
1 row in set (0.00 sec)
```

`UNCOMPRESSED_LENGTH()` foi adicionado no MySQL 4.1.1

`LAST_INSERT_ID([expr])`

Retorna o último valor gerado automaticamente que tenha sido inserido em um coluna `AUTO_INCREMENT`.

```
mysql> SELECT LAST_INSERT_ID();
        -> 195
```

O último ID que foi gerado e mantido no servidor em uma base por conexão. Isto significa que o valor que a função retona para um dado cliente é o valor `AUTO_INCREMENT` gerado mais recentemente por aquele cliente. O valor não pode ser afetado pelos outros clientes, mesmo se eles gerarem um valor `AUTO_INCREMENT` deles mesmos. Este comportamento assegura que você pode recuperar seu próprio ID sem se preocupar com a atividade de outros clientes e sem precisar de locks ou transações.

O valor de `LAST_INSERT_ID()` não é alterado se você atualizar uma coluna `AUTO_INCREMENT` de uma linha com um valor não-mágico (Isto é, um valor que não seja `NULL` e nem `0`).

Se você inserir muitos registros ao mesmo tempo com uma instrução insert, `LAST_INSERT_ID()` retorna o valor da primeira linha inserida. A razão para isto é tornar possível reproduzir facilmente a mesma intrução `INSERT` em algum outro servidor.

Se `expr` é dado com um argumento para `LAST_INSERT_ID()`, então o valor do argumento é retornado pela função e é configurado como o próximo valor para ser retornado pela `LAST_INSERT_ID()`. Isto pode ser útil para simular sequências:

Primeiro crie a tabela:

```
mysql> CREATE TABLE sequencia (id INT NOT NULL);
mysql> INSERT INTO sequencia VALUES (0);
```

Então a tabela pode ser usada para gerar sequência de números como estes:

```
mysql> UPDATE sequencia SET id=LAST_INSERT_ID(id+1);
```

Você pode gerar sequências sem chamar `LAST_INSERT_ID()`, mas a utilidade de se usar a função deste modo é que o valor ID é mantido no servidor como o último valor gerado automaticamente (seguro para multi-usurário). Você pode recuperar a nova ID como você leria qualquer valor `AUTO_INCREMENT` normal no MySQL. Por exemplo, `LAST_INSERT_ID()` (sem um argmento) retornará a nova ID. A função `mysql_insert_id()` da API C também pode ser usada para obter o valor.

Note que como `mysql_insert_id()` só é atualizado depois de instruções `INSERT` e `UPDATE`, você não pode utilizar a função da API C para recuperar o valor para `LAST_INSERT_ID(expr)` depois de executar outra instrução SQL como `SELECT` ou `SET`. Veja Seção 12.1.3.32 [`mysql_insert_id()`], Página 803.

`FORMAT(X,D)`

Formata o número `X` com um format como `'#,###,###.##'`, arredondado para `D` casas decimais, e retorna o resultado como uma string. Se `D` é `0`, o resultado não terá nehum ponto decimal ou parte fracionária:

```
mysql> SELECT FORMAT(12332.123456, 4);
        -> '12,332.1235'
mysql> SELECT FORMAT(12332.1,4);
        -> '12,332.1000'
mysql> SELECT FORMAT(12332.2,0);
        -> '12,332'
```

`VERSION()`

Retorna uma string indicando a versão do servidro MySQL:

```
mysql> SELECT VERSION();
        -> '3.23.13-log'
```

Note que se seu versão finalizar com `-log`, significa que o log está habilitado.

`CONNECTION_ID()`

Retorna a identificação (ID da thread) desta conexão. Cada conexão tem seu próprio ID único:

```
mysql> SELECT CONNECTION_ID();
        -> 23786
```

`GET_LOCK(str,temo_limite)`

Tenta conseguir uma trava com o nome dado pela string `str`, com um tempo limite de `timeout` segundos. Retorna `1` se o bloqueio foi obtido com sucesso, `0` se o tempo esgotou (por exemplo, porque outro cliente ja bloqueou o nome), ou `NULL` se uma erro ocorreu (tal como estouro de memória ou a threado tiver sido finalizada com `mysqladmin kill`). Uma trava é liberada quando você executa `RELEASE_LOCK()`, executa uma nova `GET_LOCK()`, ou a thread termina. (tanto de forma normal quanto anormal) Esta função pode ser usada para implementar bloqueio de aplicação ou para simular registros travados. Nomes são bloqueados em uma base ampla do servidor. Se um nome foi bloqueado por um cliente, `GET_LOCK()` trava qualquer pedido de bloqueio de outro cliente com o mesmo nome. Isto permite que clientes que concordam com um dado nome da trava possam usar a string para realizar travamento de consultas cooperativas:

```
mysql> SELECT GET_LOCK("lock1",10);
        -> 1
mysql> SELECT IS_FREE_LOCK("lock2");
        -> 1
mysql> SELECT GET_LOCK("lock2",10);
        -> 1
mysql> SELECT RELEASE_LOCK("lock2");
        -> 1
mysql> SELECT RELEASE_LOCK("lock1");
        -> NULL
```

Note que a segunda chamada de `RELEASE_LOCK()` retorna `NULL` porque a trava `"lock1"` foi liberada automaticamente pela segunda chamada `GET_LOCK()`.

`RELEASE_LOCK(str)`

Libera a trava nomeada pela string `str` que foi obtida com `GET_LOCK()`. Retorna `1` se a trava foi liberada, `0` se a trava não foi bloquada pela thread (caso onde a trava não é liberada), e `NULL` se o nome da trava não existe. (A trava nunca exitirá se ela nunca for obtida pela chamada de `GET_LOCK()` ou se ela ja tiver sido liberada).

A instrução `DO` é conveniente para ser utilizada com `RELEASE_LOCK()`. Veja Seção 6.4.10 [DO], Página 597.

`IS_FREE_LOCK(str)`

Verifica se a trava chamada `str` está livre para ser utilizada (ex. não está bloqueada). Retorna `1` se a trava está liver (ninguém a esta usando), `0` se a trava está em uso, e `NULL` caso ocorra erro (como argumentos incorretos).

`BENCHMARK(cont,expr)`

A função `BENCHMARK()` executa a expressão `expr` repetidamente `cont` vezes. Ela pode ser usada para medir a velocidade em que o MySQL processa a expressão.

O valor resultante é sempre `0`. A intenção é usá-la no clientei `mysql`, relatando o tempo de execução da consulta:

```
mysql> SELECT BENCHMARK(1000000,ENCODE("hello","goodbye"));
+----------------------------------------------+
| BENCHMARK(1000000,ENCODE("hello","goodbye")) |
+----------------------------------------------+
|                                            0 |
+----------------------------------------------+
1 row in set (4.74 sec)
```

O tempo relatado é o tempo decorrido no cliente, não o tempo de CPU no servidor. Pode ser aconselhável executar `BENCHMARK()` diversas vezes e interpretar o resultado cosiderado o peso da carga da maquina servidora.

`INET_NTOA(expr)`

Dado um endereço numérico de rede (4 ou 8 bytes), retorna a representacão no formato com pontos do endereço como uma string:

```
mysql> SELECT INET_NTOA(3520061480);
       ->  "209.207.224.40"
```

`INET_ATON(expr)`

Dada a represenação com pontos de um endereço de rede como uma string, retorna um inteiro que representa o valor numérico deste endereço. Endereços podem ter 4 ou 8 bytes de endereçamento:

```
mysql> SELECT INET_ATON("209.207.224.40");
       ->  3520061480
```

O número gerado é sempre na ordem de bytes da rede; por exemplo o número acima é calculado como `209*256^3 + 207*256^2 + 224*256 +40`.

`MASTER_POS_WAIT(nome_log, log_pos [, tempo_limite])`

Envia blocos o slave alcançar (ex.: ter lido e aplicado todas as atualizações) a posição específica no log master. Se a informação master não está inicializada, ou se os argumentos estão incorretos, retorna `NULL`. Se o slave não está em execução, enviará blocos e irá esperar até que ele seja iniciado e vá para (ou passe por) a posição especificada. Se o slave já passou pela posição especificada, retorna imediatamente.

Se `tempo_limite` (novo na versão 4.0.10) é especificado, irá esperar até que `tempo_limite` segundos tenham se passado. `tempo_limite` deve ser maior que 0; zero ou um `tempo_limite` negativo significa sem tempo_limite. O valor de retorno é o número de eventos de log que ele tem que esperar para obter a posição especificada, `NULL` no caso de erro, ou `-1` se o tempo_limite tiver sido excedido.

O comando é útil para controle de sincronização mo master/slave.

`FOUND_ROWS()`

Uma instrução `SELECT` pode incluir uma cláusula `LIMIT` para restringir o número de linhas que o servidor retorna para um cliente. Em alguns casos, é desejável saber quantas linhas a instrução teria retornado sem o `LIMIT`, mas sem executar a instrução novamente. Para obter esta contagem de linhas,

inclua uma opção SQL_CALC_FOUND_ROWS na instrução SELECT, então chame FOUND_ROWS() loga depois:

```
mysql> SELECT SQL_CALC_FOUND_ROWS * FROM nome_tabela
       WHERE id > 100 LIMIT 10;
mysql> SELECT FOUND_ROWS();
```

O segundo SELECT irá retornar um número indicando quantas linhas o primeiro SELECT teria retornado se ele fosse escrito sem a cláusula LIMIT. (Se o instrução SELECT anterior não inclui a opção SQL_CALC_FOUND_ROWS, então FOUND_ROWS() pode retornar um resultado diferente quando LIMIT é usado daquele que não é usado).

Note que se você estiver usando SELECT SQL_CALC_FOUND_ROWS ..., o MySQL tem que calcular quantos registros existem em todo o conjunto de resultados. No entanto, isto é mais rápido que se você não utilizar LIMIT, já que o resultado precisa ser enviado ao cliente.

SQL_CALC_FOUND_ROWS e FOUND_ROWS() podem ser úteis em situações em que você queira restringir o número de registros que uma consulta retorna, mas também determinar o número de linhas em todo o resultado sem executar a consulta novamente. Um exemplo é um script web que apresenta um display paginado contendo links para as páginas que mostram outras seções de um resultado de busca. Usar FOUND_ROWS() lhe permite determinar quantos outras páginas são necessárias para o resto do resultado.

O uso de SQL_CALC_FOUND_ROWS e FOUND_ROWS() é mais complexa para consultas UNION que para instruções SELECT simples, porque LIMIT pode ocorrer em vários lugares em um UNION. Ele pode ser aplicado a instruções SELECT individuais no UNION, ou globais ao resultado UNION como um todo.

A intenção de SQL_CALC_FOUND_ROWS para UNION é que ele deve retornar a contagem das linhas que seriam retornadas sem um LIMIT global. As consições para uso de SQL_CALC_FOUND_ROWS com UNION são:

- A palavra chave SQL_CALC_FOUND_ROWS deve aparecer na primeira SELECT do UNION.
- O valor de FOUND_ROWS() é exato apenas se UNION ALL for usado. Se UNION sem ALL for usado, as duplicatas são removidas e o valor de FOUND_ROWS() é apenas aproximado.
- Se nenhum LIMIT está presente no UNION, SQL_CALC_FOUND_ROWS é ignorado e retorna o número de linhas na tabela temporária que é criada para processar o UNION.

SQL_CALC_FOUND_ROWS e FOUND_ROWS() estão disponíveis a partir da versão 4.0.0 do MySQL.

6.3.7 Funções e Modificadores para Usar com Cláusulas GROUP BY

6.3.7.1 Funções GROUP BY

Se você utiliza um função de agrupamento em uma instrução que não contenha um cláusula GROUP BY, equivale a fazer um agrupamento com todos os registros.

COUNT(expr)

Retorna a quantidade de valores não-NULL nos registros recuperados por uma instrucao SELECT:

```
mysql> SELECT estudante.nome_estudente,COUNT(*)
    ->        FROM estudante,curso
    ->        WHERE estudante.id_estudante=curso.id_estudante
    ->        GROUP BY nome_estudante;
```

COUNT(*) difere um pouco ao retornar o número de registros recuperados, se eles possuírem ou não valores NULL.

COUNT(*) é otimizado para retornar muito rápido se SELECT recuoperar registros de uma tabela, nenhuma outra coluna for retornada, e não houver nenhuma cláusula WHERE. Por exemplo:

```
mysql> SELECT COUNT(*) FROM estudente;
```

Esta otimizacão se aplica apenas a tabelas MyISAM e ISAM, porque uma contagem exata de registros é armazenada para estes tipos de tabelas e podem ser acessadas muito rapidamente. Para mecanismos de armazenamentos transacionais (InnodB, BDB), armazenar um contagem de registros exatos é mais problemático porque múltiplas transações podem estar ocorrendo, e cada uma pode afetar a contagem.

COUNT(DISTINCT expr,[expr...])

Retorna a quantidade de regiastros com valores não-NULL diferentes:

```
mysql> SELECT COUNT(DISTINCT resultados) FROM estudente;
```

No MySQL você pode obter o número de combinação de expressões distintas que não contém NULL fornecendo uma lista de expressões. No SQL-99 você teria que concatenar todas as expressão utilizando COUNT(DISTINCT ...).

AVG(expr)

Retorna o valor médio de expr:

```
mysql> SELECT nome_estudante, AVG(nota_teste)
    ->        FROM estudante
    ->        GROUP BY nome_estudante;
```

MIN(expr)
MAX(expr)

Retorna o valor mínimo o u máximo de expr. MIN() e MAX() poder usar uma string como argumento; nestes casos eles retornam o a string de valor mínimo ou máximo. Veja Seção 5.4.3 [Índices do MySQL], Página 447.

```
mysql> SELECT nome_estudante, MIN(nota_teste), MAX(nota_teste)
    ->        FROM estudante
    ->        GROUP BY nome_estudante;
```

Em `MIN()`, `MAX()` e outras funções de agrupamento o MySQL, atualmente, compara colunas `ENUM` e `SET` pelo seu valor string em vez de fazê-lo pela sua posição relativa de string no conjunto. Isto será retificado.

`SUM(expr)`

Retorna a soma de `expr`. Note que se o conjunto de retorno não possuir registros ele retornará NULL!

`GROUP_CONCAT(expr)`

Sintaxe completa:

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
             [ORDER BY {inteiro_sem_sinal | nome_coluna | formula} [ASC |
             [SEPARATOR valor_str])
```

Esta função foi adicionada na versão 4.1 do MySQL. Ele retorna a string resultante contendo valores de um grupo:

```
mysql> SELECT nome_estudante,
    ->        GROUP_CONCAT(note_teste)
    ->        FROM estudante
    ->        GROUP BY nome_estudante;
ou
mysql> SELECT nome_estudante,
    ->        GROUP_CONCAT(DISTINCT nota_teste
    ->                     ORDER BY nota_teste DESC SEPARATOR " ")
    ->        FROM estudante
    ->        GROUP BY nome_estudante;
```

No MySQL você pode obter valores de combinações de expressões concatenados. Você pode eliminar valores duplicados utilizando `DISTINCT`. Se você quiser ordenar valores no resultado você deve utilizar a cláusula `ORDER BY`. Para ordenar inversamente, adicione a palavra chave `DESC` (descendente) ao nome da coluna que você está ordenando na cláusula `ORDER BY`. O padrão é a ordem crescente; pode-se também especificála explicitamente usando a palavra chave `ASC`. `SEPARATOR` é o valor string que deve ser inserido entre os valores no resultado. O padrão é um virgula (‘`","`’). Você pode remover o separador especificando `SEPARATOR ""`.

Você pode definir um tamanho máximo permitido com a variável `group_concat_max_len` em sua configuração. A sintaxe para se fazer isto em tempo de execução é:

```
SET [SESSION | GLOBAL] group_concat_max_len = unsigned_integer;
```

Se um tamanho máximo tiver sido atribuido, o resultado é truncado no seu tamanho máximo.

A função `GROUP_CONCAT()` é uma implementação aprimorada da função básica `LIST()` suportada pelo Sybase SQL Anywhere. `GROUP_CONCAT()` é compatível com a funcionalidade extrwemamente limitada de de `LIST()`, se utilizada em apenas uma coluna e nenhuma outra opção é especificada. `LIST()` não tem uma ordem de classificação padrão.

`VARIANCE(expr)`
: Retorna a variância padrão de `expr` (considerando linha como toda a população, não com uma amostra; assim ele tem o número de linhas como denominador). Esta é uma extensão do SQL-99 (disponível somente a partir da versão 4.1).

`STD(expr)`
`STDDEV(expr)`
: Retorna o desvio padrão de `expr` (a raiz quadrada de `VARIANCE()`). Esta é uma extensão do SQL-99. O formato `STDDEV()` desta função é fornecida para compatibilidade com Oracle.

`BIT_OR(expr)`
: Retorna o resultado da operação binária `OR` de todos os bits em `expr`. O calculo é relizado com precisão de 64 bits (`BIGINT`).

  A função retortna 0 se não houver registros coincidentes.

`BIT_XOR(expr)`
: Retorna o bitwise `XOR` de todos os bits em `expr`. O calculo é relizado com precisão de 64-bits (`BIGINT`).

  A função retorna 0 se não houver linhas coincidentes.

  Esta função está disponível a partir do MySQL 4.1.1.

`BIT_AND(expr)`
: Retorna o resultado da operação binária `AND` de todos os bits em `expr`. O calculo é relizado com precisão de 64 bits (`BIGINT`).

  A função retortna 1 se não houver registros coincidentes.

6.3.7.2 Modificadores `GROUP BY`

No MySQL 4.1.1, a cláusula `GROUP BY` permite um modificador `WITH ROLLUP` que faz com que uma linha extra seja adicionada à saida resumo. Estas linhas representam operações de resumo de nível mais alto (ou super agregadas). Assim, o `ROLLUP` permite que você responda questões em multiplos níveis de análise com uma única consulta. Ele pode ser usado, por exemplo, para fornecer suporte para operações OLAP (Online Analytical Processing - Processamento Analítico OnLine).

Como ilustração, suponha que uma tabela chamada `sales` tenha as colunas `year`, `country`, `product` e `profit` para registrar as vendas lucrativas:

```
CREATE TABLE sales
(
    year    INT NOT NULL,
    country VARCHAR(20) NOT NULL,
    product VARCHAR(32) NOT NULL,
    profit  INT
);
```

O conteúdo da tabela pode ser resumido pode ano com um simples `GROUP BY` como este:

```
mysql> SELECT year, SUM(profit) FROM sales GROUP BY year;
+------+-------------+
```

```
| year | SUM(profit) |
+------+-------------+
| 2000 |        4525 |
| 2001 |        3010 |
+------+-------------+
```

Esta saída mostra o lucro total para cada ano, mas se você também quiser determinar o lucro total somado em todos os anos, você deve adicionar os valores adicionais ou executar uma consulta adicional.

Ou você pode usar o `ROLLUP`, que fornece os dois níveis de análise com uma única consulta. Adicionando um modificador `WITH ROLLUP` a cláusula `GROUP BY` faz com que a consulta produza outra linha que mostra o total geral de todos os anos:

```
mysql> SELECT year, SUM(profit) FROM sales GROUP BY year WITH ROLLUP;
+------+-------------+
| year | SUM(profit) |
+------+-------------+
| 2000 |        4525 |
| 2001 |        3010 |
| NULL |        7535 |
+------+-------------+
```

A linha de total super-agrupada é identificada pelo valor `NULL` na coluna `year`.

`ROLLUP` tem um efeito mais complexo quando há múltiplas colunas `GROUP BY`. Neste caso, cada vez que houver um "break" (alteração no valor) em qualquer agrupamento, com exceção da última coluna, a consulta produz um linha resumo super-agrupada extra.

Por exemplo, sem `ROLLUP`, um resumo na tabela `sales` baseada no `year`, `country` e `product` pode se parecer com isto:

```
mysql> SELECT year, country, product, SUM(profit)
    -> FROM sales
        -> GROUP BY year, country, product;
+------+---------+------------+-------------+
| year | country | product    | SUM(profit) |
+------+---------+------------+-------------+
| 2000 | Finland | Computer   |        1500 |
| 2000 | Finland | Phone      |         100 |
| 2000 | India   | Calculator |         150 |
| 2000 | India   | Computer   |        1200 |
| 2000 | USA     | Calculator |          75 |
| 2000 | USA     | Computer   |        1500 |
| 2001 | Finland | Phone      |          10 |
| 2001 | USA     | Calculator |          50 |
| 2001 | USA     | Computer   |        2700 |
| 2001 | USA     | TV         |         250 |
+------+---------+------------+-------------+
```

A saída indica os valores resumidos apenas no nível year/country/product da análise. Quando `ROLLUP` é adicionado, a consulta produz diversas linhas extras:

```
mysql> SELECT year, country, product, SUM(profit)
```

```
    -> FROM sales
    -> GROUP BY year, country, product WITH ROLLUP;
+------+---------+------------+-------------+
| year | country | product    | SUM(profit) |
+------+---------+------------+-------------+
| 2000 | Finland | Computer   |        1500 |
| 2000 | Finland | Phone      |         100 |
| 2000 | Finland | NULL       |        1600 |
| 2000 | India   | Calculator |         150 |
| 2000 | India   | Computer   |        1200 |
| 2000 | India   | NULL       |        1350 |
| 2000 | USA     | Calculator |          75 |
| 2000 | USA     | Computer   |        1500 |
| 2000 | USA     | NULL       |        1575 |
| 2000 | NULL    | NULL       |        4525 |
| 2001 | Finland | Phone      |          10 |
| 2001 | Finland | NULL       |          10 |
| 2001 | USA     | Calculator |          50 |
| 2001 | USA     | Computer   |        2700 |
| 2001 | USA     | TV         |         250 |
| 2001 | USA     | NULL       |        3000 |
| 2001 | NULL    | NULL       |        3010 |
| NULL | NULL    | NULL       |        7535 |
+------+---------+------------+-------------+
```

Para esta consulta, adicionar `ROLLUP` faz com que a saída inclua uma informação resumida nos qualtro níveis de análise, não só em um. Aqui está como interpretar a saída `ROLLUP`:

- Seguindo cada conjunto de produtos para um dado ano e país, um linha de resumo extra é produzida mostrando o total para todos os produtos. Estas linhas têm a coluna `product` atribuída com `NULL`.
- Seguindo cada conjunto de linhas para um dado ano, uma l;inha resumo extra é produzida mostrando o total para todos os países e produtos. Estas linhas têm as colunas `country` e `products` atribuídas com `NULL`.
- Finalmente, seguindo todas as outras linhas, um linha resumo extra é produzida mostrando o total geral para todos os anos, países e produtos. Esta linha tem as colunas `year`, `country` e `products` atribuídas com `NULL`.

Outras Considerações ao Usar `ROLLUP`

O seguinte item lista alguns comportamentos específicaos para a implementação do `ROLLUP` no MySQL:

Quando você usa `ROLLUP`, você não pode usar uma cláusula `ORDER BY` para ordenar os resultados. (Em outras palavras, `ROLLUP` e `ORDER BY` são exclusivos mutualmente.) No entanto, você ainda tem algum controle sobre a ordem de ordenação. O `GROUP BY` no MySQL ordena os resultados, e você pode usar as palavras chaves `ASC` e `DESC` explicitamente com colunas chamadas na lista `GROUP BY` para especificar a ordem de classificação para colunas individuais. (A linha resumo de nível mais alto adicionado por `ROLLUP` ainda aparece depois da linha para as quais elas são calculadas, considerando a ordenação.)

`LIMIT` pode ser usado para restringir o númerod e linhas retornadas para o cliente. `LIMIT` é aplicado depois do `ROLLUP`, assim o limite se aplica contra as linhas extras adicionadas por `ROLLUP`. Por exemplo:

```
mysql> SELECT year, country, product, SUM(profit)
    -> FROM sales
    -> GROUP BY year, country, product WITH ROLLUP
    -> LIMIT 5;
+------+---------+------------+-------------+
| year | country | product    | SUM(profit) |
+------+---------+------------+-------------+
| 2000 | Finland | Computer   |        1500 |
| 2000 | Finland | Phone      |         100 |
| 2000 | Finland | NULL       |        1600 |
| 2000 | India   | Calculator |         150 |
| 2000 | India   | Computer   |        1200 |
+------+---------+------------+-------------+
```

Note que usar `LIMIT` com `ROLLUP` pode produzir resultados mais difíceis de interpretar, porque você têm menos contexto para entender as linhas super agrupadas.

O indicador `NULL` em cada linha super-agrupadas são produzidas quando a linha é enviada para o cliente. O servidor olha por cada coluna chamada na cláusula `GROUP BY` seguindo aquela mais a esquerda que tem o valor alterado. Para qualquer coluna no resultado com o nome que é uma combinação léxica de qualquer daqueles nomes, seu valor é definido com `NULL`. (Se você especifica o agrupamento de colunas pelo número da coluna, o servidor identifica quais colunas definir com `NULL` pelo número.)

Como os valores `NULL` em linhas super agrupadas são colocadas dentro do resultado como um estágio posterior no processamento da consulta, você não pode testá-los com valores `NULL` dentro da própria consulta. Por exemplo, você não pode adicionar `HAVING product IS NULL` a consulta para eliminar da saída todas as linhas com exceção das agrupadas.

Por outro lado, o valor `NULL` aparece como `NULL` no lado do cliente e pode ser testado usando qualquer interface de programação do cliente MySQL.

6.3.7.3 `GROUP BY` com Campos Escondidos

O MySQL tem extendido o uso de `GROUP BY`. Você pode utilizar colunas ou cálculos na expressão `SELECT` que não aparecem na parte `GROUP BY`. Ele espera por *qalquer valor possível para este grupo*. Você pode utilizar isto para conseguir um melhor desempenho evitando ordenação e agrupamento em itens desnecessários. Por exemplo, você não precisa fazer um agrupamento em `cliente.nome` na consulta seguinte:

```
mysql> SELECT pedido.idcliente,cliente.nome,MAX(pagamento)
    ->        FROM pedido, cliente
    ->        WHERE pedido.idcliente = cliente.idcliente
    ->        GROUP BY pedido.idcliente;
```

No padrão SQL, você teria que adicionar `cliente.nome` a cláusula `GROUP BY`. No MySQL, o nomê é redundante se você não o executa em modo ANSI.

Não utilize este recurso se as colunas omitidas na parte `GROUP BY` não são únicas no grupo! Você obterá resultados inexperados.

Em alguns casos, você pode utilizar `MIN` e `MAX` para obter o valor de uma coluna específica, mesmo que ele não seja único. O exemplo seguinte fornece o valor de `coluna` do registro contendo o menor valor na coluna `ordem`:

```
SUBSTR(MIN(CONCAT(RPAD(ordem,6,' '),coluna)),7)
```

Veja Seção 3.6.4 [example-Maximum-column-group-row], Página 197.

Note que se você estiver usando a versão 3.22 do MySQL (ou anterior) ou se estiver tentando seguir o SQL-99, você não pode utilizar expressões nas cláusulas `GROUP BY` or `ORDER BY`. Você pode contornar esta limitação utilizando um alias para a expressão:

```
mysql> SELECT id,FLOOR(value/100) AS val FROM nome_tabela
    ->        GROUP BY id,val ORDER BY val;
```

Na versão 3.23 do MySQL você pode fazer:

```
mysql> SELECT id,FLOOR(value/100) FROM nome_tabela ORDER BY RAND();
```

6.4 Manipulação de Dados: SELECT, INSERT, UPDATE e DELETE

6.4.1 Sintaxe SELECT

```
SELECT [STRAIGHT_JOIN]
       [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
       [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
       [DISTINCT | DISTINCTROW | ALL]
    expressão_select,...
    [INTO {OUTFILE | DUMPFILE} 'nome_arquivo' opções_exportação]
    [FROM tabelas_ref
      [WHERE definição_where]
      [GROUP BY {inteiro_sem_sinal | nome_col | formula} [ASC | DESC], ...
        [WITH ROLLUP]]
      [HAVING where_definition]
      [ORDER BY {inteiro_sem_sinal | nome_coluna | formula} [ASC | DESC], ...]
      [LIMIT [offset,] row_count | row_count OFFSET offset]
      [PROCEDURE nome_procedimento(lista_argumentos)]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

`SELECT` é utilizado para retornar registros selecionados de uma ou mais tabelas. Cada `expressão_select` indica as colunas que você deseja recuperar. `SELECT` tanbém pode ser utilizado para retornar registros calculados sem referência a nenhuma tabela. Por exemplo:

```
mysql> SELECT 1 + 1;
         -> 2
```

Todas as cláusulas usada devem ser fornecidas exatamente na ordem mostrada na descrição da sintaxe. Por exemplo, uma cláusula `HAVING` deve vir depois de qualquer cláusula `GROUP BY` e antes de qualquer cláusula `ORDER BY`.

- Uma expressão `SELECT` pode utilizar um alias usando `AS nome_alias`. O alias é usado como o nome da coluna da expressão e pode ser usado com cláusulas `ORDER BY` ou `HAVING`. Por exemplo:

```
mysql> SELECT CONCAT(primeiro_nome,' ',ultimo_nome) AS nome_completo
    FROM minha_tabela ORDER BY nome_completo;
```

  A palavra chave `AS` é opcional quando se utiliza alias em uma expressão `SELECT`. O exemplo anterior poderia ser escrito assim:

```
mysql> SELECT CONCAT(last_name,', ',first_name) full_name
    FROM mytable ORDER BY full_name;
```

  Como `AS` é opcional, pode ocorrer um problema se você esquecer a vírgula entre duas expressões `SELECT`: O MySQL interpretará o segundo como um nome de alias. Por exemplo, na seguinte instrução, `columnb` é tratada como um nome de alias:

```
mysql> SELECT columna columnb FROM mytable;
```

- Não é permitido utilizar um alias de coluna em uma cláusula `WHERE`, pois o valor da coluna pode ainda não ter sido determinado quando a cláusula `WHERE` for executada. Veja Seção A.5.4 [Problemas com alias], Página 933.
- A cláusula `FROM table_references` indica a tabela de onde os registros serão retornados. Se você indicar mais de uma tabela, você estará realizando uma join. Para informações sobre a sintaxe de join, veja Seção 6.4.1.1 [`JOIN`], Página 567. Para cada tabela especificada, você pode, opcionalmente, especificar um alias.

```
nome_tabela [[AS] alias] [[USE INDEX (lista_indice)] | [IGNORE INDEX (lista_indi
```

  Como na versão 3.23.12 do MySQL, você pode dar sugestões sobre qual índice o MySQL deve usar ao recuperar informações de uma tabela. Isto é útil se `EXPLAIN` mostrar que o MySQL esta utilizando o índice errado da lista de índices possíveis. Especificando `USE INDEX (lista_indice)` você pode dizer ao MySQL para usar somente um dos índices possíveis para encontrar registros em uma tabela. A sintaxe alternativa `IGNORE INDEX (lista_indice)` pode ser usada para dizer ao MySQL para não utilizar alguns índices particulares.

  Na versão 4.0.9 do MySQL você também pode usar `FORCE INDEX`. Ele funciona como `USE INDEX (lista_indice)` mas ele assume que uma varredura em uma tabelas é MUITO cara. Em outras palavras, uma varredura só será usada se não houver nenhum modo de utilizar um dos índices dados para encontrar registros nas tabelas.

  `USE/IGNORE/FORCE KEY` é sinônimo de `USE/IGNORE/FORCE INDEX`.

  Nota: `USE/IGNORE/FORCE INDEX` afeta apenas os índices usados quando o MySQL decide como encontrar registros na tabela e como fazer a ligação. Ele não tem efeito se um índice será usado ao resolver um `ORDER BY` ou `GROUP BY`.

  No MySQL 4.0.14 você pode usar `SET MAX_SEEKS_FOR_KEY=#` como um modo alternativo de forçar o MySQL a preferir a busca em chaves em vez de varrer a tabela.
- Você pode se referir a uma tabela como `nome_tabela` (dentro do banco de dados atual) ou como `nomebd.nome_tabela` para especificar um banco de dados. Você pode se referir a um coluna como `nome_coluna`, `nome_tabela.nome_coluna` ou `nomebd.nome_tabela.nome_coluna`. Você não precisa especificar um prefixo `nome_tabla` ou `nomebd.nome_tabela` para referência a uma coluna em uma instrução `SELECT` a menos a referência seja ambígua. Veja Seção 6.1.2 [Legal names], Página 472,

para exemplos de ambiguidade que exigem a forma mais explicita de referência a coluna.

- A partir da versão 4.1.0, você pode especificar `DUAL` como um nome de tabela dummy, em situações onde nenhuma tabela for referênciada. Este é um recurso puramente para compatibilidade, alguns outros servidores exijem esta sintaxe.

```
mysql> SELECT 1 + 1 FROM DUAL;
         -> 2
```

- Pode se definir um alias fazendo referência a uma tabela utilizando `nome_tabela [AS] nome_alias`:

```
mysql> SELECT t1.nome, t2.salario FROM funcionarios AS t1, info AS t2
    ->        WHERE t1.nome = t2.nome;
mysql> SELECT t1.nome, t2.salario FROM funcionarios t1, info t2
    ->        WHERE t1.nome = t2.nome;
```

- Colunas selecionadas para saída podem ser referidas em cláusulas `ORCER BY` e `GROUP BY` usando nomes de colunas, alias de colunas ou posições de colunas. As posições de colunas começam com 1:

```
mysql> SELECT college, region, seed FROM tournament
    ->        ORDER BY region, seed;
mysql> SELECT college, region AS r, seed AS s FROM tournament
    ->        ORDER BY r, s;
mysql> SELECT college, region, seed FROM tournament
    ->        ORDER BY 2, 3;
```

  Para ordenar inversamente, adicione a palavra-chave `DESC` (descendente) ao nome da coluna na cláusula `ORDER BY` na qual você está ordenando. A ordem padrão é ascedente; ela pode ser especificada explicitamente usando a palavra-chave `ASC`.

- Na cláusula `WHERE`, você pode usar qualquer uma das funções suportadas pelo MySQL. Exceto para funções de agruopamento (resumo) Veja Seção 6.3 [Funções], Página 504.

- A cláusula `HAVING` pode se referir a qualquer coluna ou alias definido na `expressão_select`. Ele é aplicado no final, pouco antes dos itens serem enviados ao cliente, sem otimização. `LIMIT` é aplicada depois de `HAVING`.) estar na cláusula `WHERE`. Por exemplo, não escreva isto:

```
mysql> SELECT nome_col FROM nome_tabela HAVING nome_col > 0;
```

  Escreva assim:

```
mysql> SELECT nome_col FROM nome_tabela WHERE nome_col > 0;
```

  Na versão 3.22.5 ou posterior, você também pode escrever consultar desta forma:

```
mysql> SELECT usuario,MAX(salario) FROM usuarios
    ->        GROUP BY usuario HAVING MAX(salario)>10;
```

  Em versões mais antigas, você pode escrever desta forma:

```
mysql> SELECT usuario,MAX(salario) AS soma FROM usuarios
    ->        group by usuario HAVING soma>10;
```

- As opções `DISTINCT`, `DISTINCTROW` e `ALL` especificam quando registros duplicados devem ser retornados. O padrão é (`ALL`), todos os registros coincidentes são retornados. `DISTINCT` e `DISTINCTROW` são sinônimos e espcificam que registros duplicados no conjunto de resultados devem ser remopvidos.

- `STRAIGHT_JOIN`, `HIGH_PRIORITY` e opções começando com `SQL_` são extensões do MySQL para SQL-99.
  - No MySQL 4.1.1, `GROUP BY` permite um modificador `WITH ROLLUP`. Veja Seção 6.3.7.2 [Modificadores `GROUP BY`], Página 558.
  - `HIGH_PRIORITY` dará uma prioridade maior ao `SELECT` do que para uma instrução que atualizam uma tabela. Você só deve isto para consultas que sejam rápidas e devam ser feitas imediatamente. Uma consulta `SELECT HIGH_PRIORITY` retornará se a tabela está bloqueada para leitura memsmo se houver uma instrução de atualização que estiver esperando a liberação da tabela.
  - `SQL_BIG_RESULT` pode ser usado com `GROUP BY` ou `DISTINCT` para dizer ao otimizador que o conjunto de resultados terá muitas linhas. Neste caso, o MySQL usará diretamente tabelas temporarias em disco se necessário. O MySQL também irá, neste caso, preferir ordenar fazendo uma tabela temporária com um cahve nos elementos `GROUP BY`.
  - `SQL_BUFFER_RESULT` força para que o resultado seja colocado em uma tabela temporária. Isto ajuda o MySQL a liberar as travas de tabelas mais cedo e ajudará nos casos onde ele levá muito tempo para enviar o conjunto de resultado ao cliente.
  - `SQL_SMALL_RESULT`, uma opção especifica do MySQL, pode ser usada com `GROUP BY` ou `DISTINCT` para dizer ao otimizador que o conjunto de resultados será pequeno. Neste caso, o MySQL usa tabelas temporárias rápidas para armazenar a tabela resultante em vez de usar ordenação. Na versão 3.23 do MySQL isto não é necessário normalmente.
  - `SQL_CALC_FOUND_ROWS` (versão 4.0.0 e acima) diz ao MySQL para calcular quantas linhas haveriam no conjunto de resultados, desconsiderando qualquer cláusula `LIMIT`. O número de linhas pode ser recuperado com `SELECT FOUND_ROWS()`. Veja Seção 6.3.6.2 [Funções diversas], Página 547.

    Por favor, note que em nversões anteriores a 4.1.0 isto não funciona com `LIMIT 0`, o qual é otimizado para retornar instantaneamente (resultando em 0 registros). Veja Seção 5.2.9 [Otimização LIMIT], Página 437.
  - `SQL_CACHE` diz ao MySQL para armazenar o resultado da consulta em um cache de consultas se você estiver utilizando `QUERY_CACHE_TYPE=2` (`DEMAND`). Veja Seção 6.9 [Cache de consultas], Página 625. No caso da consulta com UNIONs e/ou subqueries esta opção terá efeito se usada em qualquer SELECT da consulta.
  - `SQL_NO_CACHE` diz ao MySQL para não armazenar o resulado da consulta nesta cache de consultas. Veja Seção 6.9 [Cache de consultas], Página 625. No caso da consulta com UNIONs e/ou subqueries esta opção terá efeito se usada em qualquer SELECT da consulta.
- Se você utiliza `GROUP BY`, os registros de saída serão ordenados de acordo com o `GROUP BY` como se você tivesse um `ORDER BY` sobre todos os campos no `GROUP BY`. O MySQL tem expandido a cláusula `GROUP BY` para que você também possa especificar `ASC` e `DESC` depois das colunas chamadas na cláusula:

```
SELECT a,COUNT(b) FROM tabela_teste GROUP BY a DESC
```

- O MySQL tem extendido o uso do `GROUP BY` para lhe permitir selecionar campos que não estão mencionados na cláusula `GROUP BY`. Se você não está conseguindo os resultados

esperados ara a sua consulta, leia a descrição de `GROUP BY`. Veja Seção 6.3.7 [Group by functions and modifiers], Página 555.

- A partir do MySQL 4.1.1, `GROUP BY` permite um modificador `WITH ROLLUP`. Veja Seção 6.3.7.2 [`GROUP BY` Modifiers], Página 558.
- A cláusula `LIMIT` pode ser usada para restringir o número de linhas retornadas pela instrução `SELECT`. `LIMIT` utiliza um ou dois agumebntos numéricos, que devem ser constants inteiras.

  Com um argumento. o valor especifíca o número de linhas para retornar do iníco do resultado. Com dois argumentos, o primeiro especifica a posição do primeiro registro a ser retornado e o segundo especifica o número máximo de linhas a retornar. A posição do registro inicial é 0 (não 1):

  Para ser compatível com o PostgreeSQL, o MySQL suporta a sintaxe: `LIMIT row_count OFFSET offset`.

  ```
  mysql> SELECT * FROM tabela LIMIT 5,10;  # Recupera linhas 6-15
  ```

  Para recuperar todos os registros de um determinado offset até um fim do resultado você pode usar um número grande como segundo parâmetro:

  ```
  mysql> SELECT * FROM tabela LIMIT 95,18446744073709551615; # Recupera linhas 96-
  ```

  Se um dos argumentos é dado, ele indica o número máximo de linhas a retornar:

  ```
  mysql> SELECT * FROM tabela LIMIT 5;     # Recupera as primeiras 5 linhas
  ```

  Em outras palavras, `LIMIT n` é equivalente a `LIMIT 0,n`.
- A forma `SELECT ... INTO OUTFILE 'nome_arquivo'` do `SELECT` grava os registros selecionados em um arquivo. O arquivo é criado na máquina servidora e não pode já existir (entre outras coisas, isto previne tabelas de banco de dados e arquivos tais como '`/etc/passwd`' de serem destruídos). Você deve ter o privilégio `FILE` na máquina servidora para utilizar esta forma de `SELECT`.

  A instrução `SELECT ... INTO OUTFILE` tem como intenção deixar que você descarregue rapidamente um tabela de uma máquina servidora. Se você quiser criar o arquivo resultante em outra máquina, diferente do servidor, você não deve usar `SELECT ... INTO OUTFILE`. Neste caso você deve usar algum programa cliente como `mysqldump --tab` ou `mysql -e "SELECT..." > outfile` para gerar o arquivo.

  `SELECT ... INTO OUTFILE` é o complemento de `LOAD DATA INFILE`; a sintaxe para a parte `opções_exportação` de uma instrução consiste das mesmas cláusulas `CAMPOS` e `LINHAS` que são usadas com a instrução `LOAD DATA INFILE`. Veja Seção 6.4.8 [`LOAD DATA`], Página 588.

  No arquivo texto resultante, somente os seguintes coracteres são escritos com o caracter `ESCAPE BY`:

  - O caracter `ESCAPE BY`
  - O primeiro caracter em `FIELDS TERMINATED BY`
  - O primeiro caracter em `LINES TERMINATED BY`

  Adicionalmente, `ASCII 0` é convertido para `ESCAPE BY` seguido por 0 (`ASCII 48`).

  A razão para o mostrado acima é que você `deve` escapar qualquer caracter `FIELDS TERMINATED BY`, `ESCAPE BY`, or `LINES TERMINATED BY` para termos a segurança que o

arquivo poderá ser lido de volta. É feito escape de `ASCII 0` para facilitar a visuzlização com alguns paginadores.

Como o arquivo resultante não tem que estar em conformidade com a sintaxe SQL, nada mais precisa ser seguido de caraceres de escape.

Aqui segue um exemplo de como se obter um arquivo no formato usado por muitos programas antigos.

```
SELECT a,b,a+b INTO OUTFILE "/tmp/result.text"
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY "\n"
FROM tabela_teste;
```

- Se você utilizar `INTO DUMPFILE` em vez de `INTO OUTFILE`, o MySQL só irá escrever um linha no arquivo, sem nenhum terminador de linha ou colunas e sem realizar nenhum processo de escape. Ele é útil se você quiser armazenar um valor `BLOB` em um arquivo.
- Note que qualuqer arquivo criado por `INTO OUTFILE` e `INTO DUMPFILE` serão escritos por todos os usuários no servidor! A razão é que o servidor MySQL não pode criar um arquivo que pertence a qualquer um além do usuário que o está executando (você nunca deve executar `mysqld` como `root`). Assim o arquivo tem que poder ser gravado por todos para que você possa manipular o seu conteúdo.
- Uma cláusula `PROCEDURE` chama um procedimento que devia processar os dados em um resultado. Para um exemplo, veja Seção 14.3.1 [procedure analyse], Página 909.
- Se você utilizar `FOR UPDATE` em um mecanismo de armazenamento com locks de páginas ou registros, as linhas examinadas serão travadas para escrita até o fim da transação atual.

6.4.1.1 Sintaxe `JOIN`

O MySQL suporta as seguintes sintaxes `JOIN` para uso em instruções `SELECT`:

```
tabela_ref, tabela_ref
tabela_ref [INNER | CROSS] JOIN table_reference [join_condition]
tabela_ref STRAIGHT_JOIN tabela_ref
tabela_ref LEFT [OUTER] JOIN table_reference [join_condition]
tabela_ref NATURAL [LEFT [OUTER]] JOIN tabela_ref
{ OJ tabela_ref LEFT OUTER JOIN tabela_ref ON expr_condicional }
tabela_ref RIGHT [OUTER] JOIN table_reference [join_condition]
tabela_ref NATURAL [RIGHT [OUTER]] JOIN tabela_ref
```

Onde `tabela_ref` é definido como:

```
nome_tabela [[AS] alias] [[USE INDEX (lista_indice)] | [IGNORE INDEX (lista_indice)
```

a `condição_join` é definido como:

```
ON expr_condicional |
USING (lista_colunas)
```

Geralamente você não deverá ter nenhuma condição na parte `ON` que é usada para restringir quais registros você terá no seu resultado, mas ao invés disto, especificar estas condições na cláusula `WHERE`. Existem exceções para isto.

Note que a sintaxe `INNER JOIN` permite uma `condição_join` apenas a partir da versão 3.23.17. O mesmo acontece para `JOIN` e `CROSS JOIN` apenas a partir do MySQL 4.0.11.

A última sintaxe `LEFT OUTER JOIN` mostrada na lista anterior só existe para compatibilidade com ODBC:

- Pode se usar um alias para referência a tabelas com `nome_tabela AS nome_alias` ou `nome_tabela nome_alias`:

```
mysql> SELECT t1.nome, t2.salario FROM funcionarios AS t1, info AS t2
    ->        WHERE t1.nome = t2.nome;
```

- A condicional `ON` é qualquer condição da forma que pode ser usada em uma cláusula `WHERE`.
- Se não houver registros coincidentes para a tabela a direita da parte `ON` ou `USING` em um `LEFT JOIN`, uma linha com `NULL` atribuído a todas as colunas é usada para a tabela a direita. Você pode usar este fato para encontrar registro em uma tabela que não houver contrapartes em outra tabela

```
mysql> SELECT tabela1.* FROM tabela1
    ->        LEFT JOIN tabela2 ON tabela1.id=tabela2.id
    ->        WHERE tabela2.id IS NULL;
```

  Este exemplo encontra todas as linhas em `tabela1` com um valor `id` que não está presente em `tabela2` (isto é, toda as linhas em `tabela1` sem linha correspondente em `tabela2`). Assume-se que `tabela2.id` é declarada `NOT NULL`. Veja Seção 5.2.7 [Otimização LEFT JOIN], Página 435.
- A cláusula `USING (lista_colunas)` nomeia uma lista de colunas que devem existir em ambas as tabelas. As seguintes duas cláusulas são semanticamente idênticas:

```
a LEFT JOIN b USING (c1,c2,c3)
a LEFT JOIN b ON a.c1=b.c1 AND a.c2=b.c2 AND a.c3=b.c3
```

- Um `NATURAL [LEFT] JOIN` de duas tabelas é definido para ser semanticamente equivalente a um `INNER JOIN` ou um `LEFT JOIN` com uma cláusula `USING` que nomeia todas as colunas que exitem em ambas as tabelas.
- `INNER JOIN` e `,` (vírgula) são semanticamente equivalentes na ausência da condição join: ambos produzirão um produto Cartesiano entre as tabelas especificadas. (isto é, todos os registros na primeira tabela serão ligados com todos os registros na segunda tabela).
- `RIGHT JOIN` funciona de forma análoga a um `LEFT JOIN`. Para manter o código portável entre banco de dados, é recomendado usar `LEFT JOIN` em vez de `RIGHT JOIN`.
- `STRAIGHT_JOIN` é identico a `JOIN`, exceto pelo fato de que a tabela de esquerda sempre é lida antes da tabela da direita. Ele pode ser usado para aqueles casos (poucos) onde o otimizador join coloca as tabelas na ordem errada.
- Como na versão 3.23.12, você pode dar sugestões sobre qual índice o MySQL deve us quando retornar informações de uma tabela. Isto é útil se `EXPLAIN` mostar que o MySQL está utilizando o índice errado da lista de índices possíveis. Especificando `USE INDEX (lista_indice)`, você pode dizer ao MySQL para usar somente um dos índices possíveis para encontrar registros em uma tabela. A sintaxe alternativa `IGNORE INDEX (lista_indice)` pode ser usado para dizer ao MySQL para não utilizar índices particulares.

Na versão 4.0.9 do MySQL você também pode utilizar `FORCE INDEX`. Ele funciona como `USE INDEX (key_list)` mas com assume que uma varredura na tabela é MUITO cara. Em outras palavras, uma varredura na tabela só será feita se não houver modo de uitlizar um dos índices fornecidos para se enecontrar registros no tabela.

`USE/IGNORE KEY` são sinônimos de `USE/IGNORE INDEX`.

Nota: `USE/IGNORE/FORCE INDEX` afeta apenas os índices usados quando o MySQL decide como encontrar registros na tabela e como fazer a ligação. Ele não tem efeito se um índice será usado ao resolver um `ORDER BY` ou `GROUP BY`.

Alguns exemplos:

```
mysql> SELECT * FROM tabela1,tabela2 WHERE tabela1.id=tabela2.id;
mysql> SELECT * FROM tabela1 LEFT JOIN tabela2 ON tabela1.id=tabela2.id;
mysql> SELECT * FROM tabela1 LEFT JOIN tabela2 USING (id);
mysql> SELECT * FROM tabela1 LEFT JOIN tabela2 ON tabela1.id=tabela2.id
    ->          LEFT JOIN tabela3 ON tabela2.id=tabela3.id;
mysql> SELECT * FROM tabela1 USE INDEX (chave1,chave2)
    ->          WHERE chave1=1 AND chave2=2 AND chave3=3;
mysql> SELECT * FROM tabela1 IGNORE INDEX (chave3)
    ->          WHERE chave1=1 AND chave2=2 AND chave3=3;
```

Veja Seção 5.2.7 [Otimização `LEFT JOIN`], Página 435.

6.4.1.2 Sintaxe `UNION`

```
SELECT ...
UNION [ALL]
SELECT ...
  [UNION
   SELECT ...]
```

`UNION` foi implementado no MySQL 4.0.0.

`UNION` é usado para combinar o resultado de muitas instruções `SELECT` em um único conjunto de resultados.

As colunas listadas na porção expressão_select de `SELECT` devem ter o mesmo tipo. Os nomes das colunas usadas na primeira consulta `SELECT` serão usadas como nomes de colunas para o resultado retornado.

Os comandos `SELECT` são comandos selects normais, mas com a seguinte restrição:

- Somente o último comando `SELECT` pode ter `INTO OUTFILE`.

Se você não utilzar a palavra-chave `ALL` para o `UNION`, todas as linhas retornadas serão únicas, como se você tivesse utilizado um `DISTINCT` para o resultado final. Se você especificar `ALL`, você obterá todos os regitros encontrados em todas as instruções `SELECT`.

Se você quiser usar um `ORDER BY` para o resultado `UNION` final, você deve utilizar parenteses:

```
(SELECT a FROM nome_tabela WHERE a=10 AND B=1 ORDER BY a LIMIT 10)
UNION
(SELECT a FROM nome_tabela WHERE a=11 AND B=2 ORDER BY a LIMIT 10)
ORDER BY a;
```

6.4.2 Sintaxe de Subquery

Uma subquery é uma instrução `SELECT` dentro de outra instrução. Por exemplo:

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

No exemplo acima, `SELECT * FROM t1 ...` é a *consulta principal* (ou *instrução principal*), e `(SELECT column1 FROM t2)` é a *subquery*. Dizemos que a subquery está *aninhada* na consulta principal, e de fato é possível aninhar subqueries dentro de outras subqueries, a uma grande profundidade. uma subquery deve estar sempres dentro de parenteses.

A partir da versão 4.1. o MySQL suporta todas as formas de subqueries e operações que o padrão SQL exige, assim como alguns recursos que são especificos do MySQL. A principal vantagem das subqueries são:

- elas permitem consultas que estão *estruturadas* assim é possível isolar cada parte de uma instrução,
- elas fornecem modos alternativos de realizar operações que, de outra forma, exigiriam joins e unions complexos,
- elas são, na opinião de muitas pessoas, legíveis. De fato, foi a inovação das subqueries que deu às pessoas a idéia original do nome `SQL` "Structured Query Language".

Com versões MySQL anteriores era necessário evitar ou contornar as subqueries, mas as pessoas que começam a escrever código agora descobrirão que subqueries são uma parte muito útil do pacote de ferramentas.

Aqui está uma instrução exemplo que mostra o ponto principal sobre a sintaxe de subquery como especificado pelo SQL padrão e suportado no MySQL.

```
DELETE FROM t1
WHERE s11 > ANY
 (SELECT COUNT(*) /* no hint */ FROM t2
 WHERE NOT EXISTS
  (SELECT * FROM t3
   WHERE ROW(5*t2.s1,77)=
    (SELECT 50,11*s1 FROM t4 UNION SELECT 50,77 FROM
     (SELECT * FROM t5) AS t5)));
```

Para as versões do MySQL anteriores a 4.1, a maioria da subqueries podem ser reescritas com sucesso usando join e outros métodos. Veja Seção 6.4.2.11 [Rewriting subqueries], Página 578.

6.4.2.1 A Subquery como um Operando Escalar

Na sua forma mais simples (a subquery *scalar* é o oposto das subqueries de *row* ou *table* que será discutido posteriormente), uma subqquery é um opernado simples. Assim você pode usá-la se um valor de uma coluna ou literal é permitido, e você pode esperar que eles tenham certas características que todos os operandos possuem: um tipo de dados, um tamanho, um indicador para informar se ele pode ser `NULL`, etc. Por exemplo:

```
CREATE TABLE t1 (s1 INT, s2 CHAR(5) NOT NULL);
SELECT (SELECT s2 FROM t1);
```

The subquery in the above SELECT has a data type of CHAR, a length of 5, a character set and collation equal to the defaults in effect at CREATE TABLE time, and an indication that the value in the column can be NULL. In fact almost all subqueries can be NULL, because if the table is empty – as in the example – then the value of the subquery will be NULL. There are few restrictions.

- A subquery's outer statement can be any one of: SELECT, INSERT, UPDATE, DELETE, SET, or DO.
- A subquery can contain any of the keywords or clauses that an ordinary SELECT can contain: DISTINCT, GROUP BY, ORDER BY, LIMIT, joins, hints, UNIONs, comments, functions, and so on.

So, when you see examples in the following sections that contain the rather Spartan construct (SELECT column1 FROM t1), imagine that your own code will contain much more diverse and complex constructions.

For example, suppose we make two tables:

```
CREATE TABLE t1 (s1 INT);
INSERT INTO t1 VALUES (1);
CREATE TABLE t2 (s1 INT);
INSERT INTO t2 VALUES (2);
```

Then perform a SELECT:

```
SELECT (SELECT s1 FROM t2) FROM t1;
```

The result will be 2 because there is a row in t2, with a column s1, with a value of 2.

The subquery may be part of an expression. If it is an operand for a function, don't forget the parentheses. For example:

```
SELECT UPPER((SELECT s1 FROM t1)) FROM t2;
```

6.4.2.2 Comparações Usando Subquery

The most common use of a subquery is in the form:

```
<non-subquery operand> <comparison operator> (<subquery>)
```

Where <comparison operator> is one of:

```
= > < >= <= <>
```

For example:

```
... 'a' = (SELECT column1 FROM t1)
```

At one time the only legal place for a subquery was on the right side of a comparison, and you might still find some old DBMSs which insist on that.

Here is an example of a common-form subquery comparison which you can't do with a join: find all the values in table t1 which are equal to a maximum value in table t2.

```
SELECT column1 FROM t1
       WHERE column1 = (SELECT MAX(column2) FROM t2);
```

Here is another example, which again is impossible with a join because it involves aggregating for one of the tables: find all rows in table t1 which contain a value which occurs twice.

```
SELECT * FROM t1
       WHERE 2 = (SELECT COUNT(column1) FROM t1);
```

6.4.2.3 Subqueries with ANY, IN, and SOME

Syntax:

```
<operand> <comparison operator> ANY (<subquery>)
<operand> IN (<subquery>)
<operand> <comparison operator> SOME (<subquery>)
```

The word ANY, which must follow a comparison operator, means "return TRUE if the comparison is TRUE for ANY of the rows that the subquery returns." For example,

```
SELECT s1 FROM t1 WHERE s1 > ANY (SELECT s1 FROM t2);
```

Suppose that there is a row in table t1 containing {10}. The expression is TRUE if table t2 contains {21,14,7} because there is a value in t2 – 7 – which is less than 10. The expression is FALSE if table t2 contains {20,10}, or if table t2 is empty. The expression is UNKNOWN if table t2 contains {NULL,NULL,NULL}.

The word IN is an alias for = ANY. Thus these two statements are the same:

```
SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 IN    (SELECT s1 FROM t2);
```

The word SOME is an alias for ANY. Thus these two statements are the same:

```
SELECT s1 FROM t1 WHERE s1 <> ANY  (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 <> SOME (SELECT s1 FROM t2);
```

Use of the word SOME is rare, but the above example shows why it might be useful. The English phrase "a is not equal to any b" means, to most people's ears, "there is no b which is equal to a" – which isn't what is meant by the SQL syntax. By using <> SOME instead, you ensure that everyone understands the true meaning of the query.

6.4.2.4 Subqueries with ALL

Syntax:

```
<operand> <comparison operator> ALL (<subquery>)
```

The word ALL, which must follow a comparison operator, means "return TRUE if the comparison is TRUE for ALL of the rows that the subquery returns". For example,

```
SELECT s1 FROM t1 WHERE s1 > ALL (SELECT s1 FROM t2);
```

Suppose that there is a row in table t1 containing {10}. The expression is TRUE if table t2 contains {-5,0,+5} because all three values in t2 are less than 10. The expression is FALSE if table t2 contains {12,6,NULL,-100} because there is a single value in table t2 – 12 – which is greater than 10. The expression is UNKNOWN if table t2 contains {0,NULL,1}.

Finally, if table t2 is empty, the result is TRUE. You might think the result should be UNKNOWN, but sorry, it's TRUE. So, rather oddly,

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT s1 FROM t2);
```

is TRUE when table t2 is empty, but

```
SELECT * FROM t1 WHERE 1 > (SELECT s1 FROM t2);
```

is UNKNOWN when table t2 is empty. In addition,

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT MAX(s1) FROM t2);
```

is UNKNOWN when table t2 is empty. In general, *tables with NULLs* and *empty tables* are *edge cases* – when writing subquery code, always consider whether you have taken those two possibilities into account.

6.4.2.5 Correlated Subqueries

A *correlated subquery* is a subquery which contains a reference to a column which is also in the outer query. For example:

```
SELECT * FROM t1 WHERE column1 = ANY
       (SELECT column1 FROM t2 WHERE t2.column2 = t1.column2);
```

Notice, in the example, that the subquery contains a reference to a column of t1, even though the subquery's FROM clause doesn't mention a table t1. So MySQL looks outside the subquery, and finds t1 in the outer query.

Suppose that table t1 contains a row where column1 = 5 and column2 = 6; meanwhile table t2 contains a row where column1 = 5 and column2 = 7. The simple expression ... WHERE column1 = ANY (SELECT column1 FROM t2) would be TRUE, but in this example the WHERE clause within the subquery is FALSE (because 7 <> 5), so the subquery as a whole is FALSE.

Scoping rule: MySQL evaluates from inside to outside. For example:

```
SELECT column1 FROM t1 AS x
  WHERE x.column1 = (SELECT column1 FROM t2 AS x
    WHERE x.column1 = (SELECT column1 FROM t3 WHERE x.column2 = t3.column1));
```

In the above, x.column2 must be a column in table t2 because SELECT column1 FROM t2 AS x ... renames t2. It is not a column in table t1 because SELECT column1 FROM t1 ... is an outer query which is *further out*.

For subqueries in HAVING or ORDER BY clauses, MySQL also looks for column names in the outer select list.

MySQL's unofficial recommendation is: avoid correlation because it makes your queries look more complex, and run more slowly.

6.4.2.6 EXISTS and NOT EXISTS

If a subquery returns any values at all, then EXISTS <subquery> is TRUE, and NOT EXISTS <subquery> is FALSE. For example:

```
SELECT column1 FROM t1 WHERE EXISTS (SELECT * FROM t2);
```

Traditionally an EXISTS subquery starts with SELECT * but it could begin with SELECT 5 or SELECT column1 or anything at all – MySQL ignores the SELECT list in such a subquery, so it doesn't matter.

For the above example, if t2 contains any rows, even rows with nothing but NULL values, then the EXISTS condition is TRUE. This is actually an unlikely example, since almost always a [NOT] EXISTS subquery will contain correlations. Here are some more realistic examples.

Example: What kind of store is present in one or more cities?

```
SELECT DISTINCT store_type FROM Stores
  WHERE EXISTS (SELECT * FROM Cities_Stores
                WHERE Cities_Stores.store_type = Stores.store_type);
```

Example: What kind of store is present in no cities?

```
SELECT DISTINCT store_type FROM Stores
  WHERE NOT EXISTS (SELECT * FROM Cities_Stores
                    WHERE Cities_Stores.store_type = Stores.store_type);
```

Example: What kind of store is present in all cities?

```
SELECT DISTINCT store_type FROM Stores S1
  WHERE NOT EXISTS (
    SELECT * FROM Cities WHERE NOT EXISTS (
      SELECT * FROM Cities_Stores
       WHERE Cities_Stores.city = Cities.city
       AND Cities_Stores.store_type = Stores.store_type));
```

The last example is a double-nested `NOT EXISTS` query – it has a `NOT EXISTS` clause within a `NOT EXISTS` clause. Formally, it answers the question "does a city exist with a store which is not in Stores?". But it's easier to say that a nested `NOT EXISTS` answers the question "is x TRUE for all y?".

6.4.2.7 Row Subqueries

The discussion to this point has been of *column (or scalar) subqueries* – subqueries which return a single column value. A *row subquery* is a subquery variant that returns a single row value – and may thus return more than one column value. Here are two examples:

```
SELECT * FROM t1 WHERE (1,2) = (SELECT column1, column2 FROM t2);
SELECT * FROM t1 WHERE ROW(1,2) = (SELECT column1, column2 FROM t2);
```

The queries above are both `TRUE` if table `t2` has a row where `column1 = 1` and `column2 = 2`.

The expression `(1,2)` is sometimes called a *row constructor* and is legal in other contexts too. For example

```
SELECT * FROM t1 WHERE (column1,column2) = (1,1);
```

is equivalent to

```
SELECT * FROM t1 WHERE column1 = 1 AND column2 = 1;
```

The normal use of row constructors, though, is for comparisons with subqueries that return two or more columns. For example, this query answers the request: "find all rows in table t1 which are duplicated in table t2":

```
SELECT column1,column2,column3
       FROM t1
       WHERE (column1,column2,column3) IN
             (SELECT column1,column2,column3 FROM t2);
```

6.4.2.8 Subqueries in the FROM clause

Subqueries are legal in a SELECT statement's FROM clause. The syntax that you'll actually see is:

```
SELECT ... FROM (<subquery>) AS <name> ...
```

The AS <name> clause is mandatory, because any table in a FROM clause must have a name. Any columns in the <subquery> select list must have unique names. You may find this syntax described elsewhere in this manual, where the term used is "derived tables".

For illustration, assume you have this table:

```
CREATE TABLE t1 (s1 INT, s2 CHAR(5), s3 FLOAT);
```

Here's how to use the *Subqueries in the FROM clause* feature, using the example table:

```
INSERT INTO t1 VALUES (1,'1',1.0);
INSERT INTO t1 VALUES (2,'2',2.0);
SELECT sb1,sb2,sb3
       FROM (SELECT s1 AS sb1, s2 AS sb2, s3*2 AS sb3 FROM t1) AS sb
       WHERE sb1 > 1;
```

Result: 2, '2', 4.0.

Here's another example: Suppose you want to know the average of the sum for a grouped table. This won't work:

```
SELECT AVG(SUM(column1)) FROM t1 GROUP BY column1;
```

But this query will provide the desired information:

```
SELECT AVG(sum_column1)
       FROM (SELECT SUM(column1) AS sum_column1
             FROM t1 GROUP BY column1) AS t1;
```

Notice that the column name used within the subquery (sum_column1) is recognized in the outer query.

At the moment, subqueries in the FROM clause cannot be correlated subqueries.

6.4.2.9 Subquery Errors

There are some new error returns which apply only to subqueries. This section groups them together because reviewing them will help remind you of some points.

- ```
  ERROR 1235 (ER_NOT_SUPPORTED_YET)
  SQLSTATE = 42000
  Message = "This version of MySQL doesn't yet support
  'LIMIT & IN/ALL/ANY/SOME subquery'"
  ```

  This means that

  ```
  SELECT * FROM t1 WHERE s1 IN (SELECT s2 FROM t2 ORDER BY s1 LIMIT 1)
  ```

  will not work, but only in some early versions, such as MySQL 4.1.1.

-
  ```

```
ERROR 1240 (ER_CARDINALITY_COL)
SQLSTATE = 21000
Message = "Operand should contain 1 column(s)"
```

This error will occur in cases like this:

```
SELECT (SELECT column1, column2 FROM t2) FROM t1;
```

It's okay to use a subquery that returns multiple columns, if the purpose is comparison. Veja Seção 6.4.2.7 [Row subqueries], Página 574. But in other contexts the subquery must be a scalar operand.

-

```
ERROR 1241 (ER_SUBSELECT_NO_1_ROW)
SQLSTATE = 21000
Message = "Subquery returns more than 1 row"
```

This error will occur in cases like this:

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

but only when there is more than one row in `t2`. That means this error might occur in code that has been working for years, because somebody happened to make a change which affected the number of rows that the subquery can return. Remember that if the object is to find any number of rows, not just one, then the correct statement would look like this:

```
SELECT * FROM t1 WHERE column1 = ANY (SELECT column1 FROM t2);
```

-

```
Error 1093 (ER_UPDATE_TABLE_USED)
SQLSTATE = HY000
Message = "You can't specify target table 'x' for update in FROM clause"
```

This error will occur in cases like this:

```
UPDATE t1 SET column2 = (SELECT MAX(column1) FROM t1);
```

It's okay to use a subquery for assignment within an `UPDATE` statement, since subqueries are legal in `UPDATE` and in `DELETE` statements as well as in `SELECT` statements. However, you cannot use the same table, in this case table `t1`, for both the subquery's `FROM` clause and the update target.

Usually, failure of the subquery causes the entire statement to fail.

6.4.2.10 Optimizing Subqueries

Development is ongoing, so no optimization tip is reliable for the long term. Some interesting tricks that you might want to play with are:

- Using subquery clauses which affect the number or order of the rows in the subquery, for example

```
SELECT * FROM t1 WHERE t1.column1 IN
  (SELECT column1 FROM t2 ORDER BY column1);
SELECT * FROM t1 WHERE t1.column1 IN
  (SELECT DISTINCT column1 FROM t2);
SELECT * FROM t1 WHERE EXISTS
```

```
  (SELECT * FROM t2 LIMIT 1);
```

- Replacing a join with a subquery, for example

```
SELECT DISTINCT column1 FROM t1 WHERE t1.column1 IN (
  SELECT column1 FROM t2);
```

  instead of

```
SELECT DISTINCT t1.column1 FROM t1, t2
  WHERE t1.column1 = t2.column1;
```

- Moving clauses from outside to inside the subquery, for example:

```
SELECT * FROM t1
  WHERE s1 IN (SELECT s1 FROM t1 UNION ALL SELECT s1 FROM t2);
```

  instead of

```
SELECT * FROM t1
  WHERE s1 IN (SELECT s1 FROM t1) OR s1 IN (SELECT s1 FROM t2);
```

  Para outro exemplo:

```
SELECT (SELECT column1 + 5 FROM t1) FROM t2;
```

  em vez de:

```
SELECT (SELECT column1 FROM t1) + 5 FROM t2;
```

- Using a row subquery instead of a correlated subquery, for example:

```
SELECT * FROM t1
  WHERE (column1,column2) IN (SELECT column1,column2 FROM t2);
```

  instead of

```
SELECT * FROM t1
  WHERE EXISTS (SELECT * FROM t2 WHERE t2.column1=t1.column1
  AND t2.column2=t1.column2);
```

- Using `NOT (a = ANY (...))` rather than `a <> ALL (...)`.
- Using `x = ANY (table containing {1,2})` rather than `x=1 OR x=2`.
- Using `= ANY` rather than `EXISTS`

The above tricks may cause programs to go faster or slower. Using MySQL facilities like the `BENCHMARK()` function, you can get an idea about what helps in your own situation. Don't worry too much about transforming to joins except for compatibility with older versions.

Some optimizations that MySQL itself will make are:

1. MySQL will execute non-correlated subqueries only once, (use `EXPLAIN` to make sure that a given subquery really is non-correlated),
2. MySQL will rewrite `IN/ALL/ANY/SOME` subqueries in an attempt to take advantage of the possibility that the select-list columns in the subquery are indexed,
3. MySQL will replace subqueries of the form

   ```
   ... IN (SELECT indexed_column FROM single_table ...)
   ```

   with an index-lookup function, which `EXPLAIN` will describe as a special join type,
4. MySQL will enhance expressions of the form

```
value {ALL|ANY|SOME} {> | < | >= | <=} (non-correlated subquery)
```

with an expression involving `MIN` or `MAX` (unless `NULL`s or empty sets are involved). For example,

```
WHERE 5 > ALL (SELECT x FROM t)
```

might be treated as

```
WHERE 5 > (SELECT MAX(x) FROM t)
```

There is a chapter titled "How MySQL Transforms Subqueries" in the MySQL Internals Manual, which you can find by downloading the MySQL source package and looking for a file named '`internals.texi`'.

6.4.2.11 Rewriting Subqueries for Earlier MySQL Versions

Up to version 4.0, only nested queries of the form `INSERT ... SELECT ...` and `REPLACE ... SELECT ...` are supported. The `IN()` construct can be used in other contexts.

It is often possible to rewrite a query without a subquery:

```
SELECT * FROM t1 WHERE id IN (SELECT id FROM t2);
```

This can be rewritten as:

```
SELECT t1.* FROM t1,t2 WHERE t1.id=t2.id;
```

The queries:

```
SELECT * FROM t1 WHERE id NOT IN (SELECT id FROM t2);
SELECT * FROM t1 WHERE NOT EXISTS (SELECT id FROM t2 WHERE t1.id=t2.id);
```

Can be rewritten as:

```
SELECT table1.* FROM table1 LEFT JOIN table2 ON table1.id=table2.id
                                        WHERE table2.id IS NULL;
```

A `LEFT [OUTER] JOIN` can be faster than an equivalent subquery because the server might be able to optimise it better – a fact that is not specific to MySQL Server alone. Prior to SQL-92, outer joins did not exist, so subqueries were the only way to do certain things in those bygone days. Today, MySQL Server and many other modern database systems offer a whole range of outer joins types.

For more complicated subqueries you can often create temporary tables to hold the subquery. In some cases, however, this option will not work. The most frequently encountered of these cases arises with `DELETE` statements, for which standard SQL does not support joins (except in subqueries). For this situation there are three options available:

- The first option is to upgrade to MySQL version 4.1.
- The second option is to use a procedural programming language (such as Perl or PHP) to submit a `SELECT` query to obtain the primary keys for the records to be deleted, and then use these values to construct the `DELETE` statement (`DELETE FROM ... WHERE ... IN (key1, key2, ...)`).
- The third option is to use interactive SQL to construct a set of `DELETE` statements automatically, using the MySQL extension `CONCAT()` (in lieu of the standard `||` operator). For example:

```
SELECT CONCAT('DELETE FROM tab1 WHERE pkid = ', "'", tab1.pkid, "'", ';')
  FROM tab1, tab2
 WHERE tab1.col1 = tab2.col2;
```

You can place this query in a script file and redirect input from it to the `mysql` command-line interpreter, piping its output back to a second instance of the interpreter:

```
shell> mysql --skip-column-names mydb < myscript.sql | mysql mydb
```

MySQL Server 4.0 supports multiple-table `DELETE`s that can be used to efficiently delete rows based on information from one table or even from many tables at the same time. Multiple-table `UPDATE`s are also supported from version 4.0.

6.4.3 Sintaxe `INSERT`

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] nome_tabela [(nome_coluna,...)]
    VALUES ((expressão | DEFAULT),...),(...),...
    [ ON DUPLICATE KEY UPDATE nome_coluna=expressão, ... ]
or  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] nome_tabela [(nome_coluna,...)]
    SELECT ...
or  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] nome_tabela
    SET nome_coluna=(expressão | DEFAULT), ...
    [ ON DUPLICATE KEY UPDATE nome_coluna=expressão, ... ]
```

`INSERT` insere novos registros em uma tabela existente. A forma `INSERT ... VALUES` da instrução insere registros baseado em valores especificados explicitamente. A forma `INSERT ... SELECT` insere linhas selecionadas de outra(s) tabela(s). A forma `INSERT ... VALUES` com listas de múltiplos valores é suportado a partir da versão 3.22.5. A sintaxe `nome_coluna=expressão` é suportada a partir da verão 3.22.10 do MySQL.

`nome_tabela` é a tabela na qual as linhas serão inseridas. A lista de nome das colunas ou a cláusula `SET` indica para quais colunas a instrução especifica valor:

- Se você não especificar a lista de colunas para `INSERT ... VALUES` ou `INSERT ... SELECT`, os valores para todas as colunas na tabela devem ser fornecidos na lista `VALUES()` ou pelo `SELECT`. Se você não souber a ordem das colunas nas tabelas, use `DESCRIBE nome_tabela` para descobrir.
- Qualquer coluna que não tiver o valor fornecido explicitamente assumirá o seu valor padrão. Por exemplo, se você especificar uma lista de colunas que não definem todas as coolunas na tabela, às colunas não definidas serão atribuídos o seu valor padrão. Atribuição de valor padrão é definido em Seção 6.5.3 [`CREATE TABLE`], Página 598.

  Você também pode utilizar a palavra-chave `DEFAULT` para atribuir o valor padrão a uma coluna (Novo na versão 4.0.3. do MySQL). Fica mais fácil de se escrever instruções `INSERT` que atribuem valor a apenas algumas colunas porque ele permite que você evite escrever uma lista `VALUES()` incompleta (uma lista que não inclu um valor para cada

coluna da tabela). De outa forma, você teria que escrever a lista de nomes de colunas correspondentes a cada valor na lista `VALUES()`.

MySQL sempre tem uma valor padrão para todos os campos. Isto é algo imposto pelo MySQL para estar apto a funcionar com tabelas transacionais e não transcaionais.

Nossa visão é que a verificação do conteúdo dos campos deve ser feita pela application and not in the database server.

- Uma `expressão` pode se referir a qualquer coluna que tenha sida definaida anteriormente na lista de valores. Por exemplo, você pode dizer:

```
mysql> INSERT INTO nome_tabela (col1,col2) VALUES(15,col1*2);
```

  Mas não:

```
mysql> INSERT INTO nome_tabela (col1,col2) VALUES(col2*2,15);
```

- Se você especificar a palavra chave `DELAYED`, o servidor coloca a linha ou linhas a serem inseridas em um buffer, e o cliente que envia a instrução `INSERT DELAYED` então pode contiuar. Se a tabela está ocupada, o servidor guarda a linha. Quando a tabela fica livre, ele começa a inserir linhas, verificando periódicamente para ver se há novos pedidos de leitura para a tabela. Se houver, a fila de linhas atrasadas é suspensa até que a tabela fique livre de novo.
- Se você especificar a palavra-chave `LOW_PRIORITY`, a execução do `INSERT` é atrasada até que nenhum outro cliente esteja lendo a tabela. Isto inclui outros clientes que começam a ler enquanto clientes existentes já estão lendo e enquanto a instrução `INSERT LOW_PRIORITY` está esperando. É possível, consequentemente, para um cliente que envia uma instrução `INSERT LOW_PRIORITY` esperar por um tempo muito longo (ou mesmo para sempre) em um ambiente de muita leitura. (É diferente de `INSERT DELAYED`, que deixa o cliente continuar de uma vez. Veja Seção 6.4.3.2 [INSERT DELAYED], Página 582. Note que `LOW_PRIORITY` não deve normalmente ser usado com tabelas `MyISAM` ja que elas disabilitam inserções concorrentes. Veja Seção 7.1 [MyISAM], Página 631.
- Se você especificar a palavra-chave `IGNORE` em um `INSERT` com muitas linhas, quqlquer linha que duplicar uma chave `PRIMARY` ou `UNIQUE` existente em uma tabela são ignorados e não são inseridos. Se você não especificar `IGNORE`, a inserção é abortada se houver quqlquer linhq que duplique um valor de uma chave existente. Você pode determinar com função `mysql_info()` da API C quantas linhas foram inseridas nas tabelas.
- Se você especificar se uma cláusula `ON DUPLICATE KEY UPDATE` (noca no MySQL 4.1.0), e uma linha que causasse a duplicação de um valor fosse inserida em uma chave `PRIMARY` ou `UNIQUE`, um `UPDATE` da linha antiga seria realizado. Por exemplo, o comando:

```
mysql> INSERT INTO table (a,b,c) VALUES (1,2,3)
    -> ON DUPLICATE KEY UPDATE c=c+1;
```

  no caso da coluna `a` ser declarada como `UNIQUE` e ja existir o valor `1`, o exemplo acima seria idêntico a

```
mysql> UPDATE table SET c=c+1 WHERE a=1;
```

  Nota: se a coluna `b` também for única, o comando `UPDATE` seria escrito como

```
mysql> UPDATE table SET c=c+1 WHERE a=1 OR b=2 LIMIT 1;
```

  e se `a=1 OR b=2` casasse com diversas linhas, somente **uma** linha será atualizada! em geral, deve-se tentar evitar utilizar a cláusula `ON DUPLICATE KEY` em tabelas com múltiplas chaves `UNIQUE`.

Desde o MySQL 4.1.1 pode-se utilizar a função `VALUES(nome_coluna)` para se referir ao valor da coluna na parte `INSERT` do comando `INSERT ... UPDATE` - que é o valor que seria inserido se não houvesse conflitos de chaves duplicadas. Esta função é especialmente útil em inserções de múltiplas linhas. Naturalmente a função `VALUES()` só tem sentido em um comando `INSERT ... UPDATE` e retorna `NULL` no caso de outros comandos.

Exemplo:

```
mysql> INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6)
    -> ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);
```

O camondo acima e idêntico a

```
mysql> INSERT INTO table (a,b,c) VALUES (1,2,3)
    -> ON DUPLICATE KEY UPDATE c=3;
mysql> INSERT INTO table (a,b,c) VALUES (4,5,6)
    -> ON DUPLICATE KEY UPDATE c=9;
```

A utilizar `ON DUPLICATE KEY UPDATE`, a opção `DELAYED` é ignorada.

- Se MySQL foi configurado usando a opção `DONT_USE_DEFAULT_FIELDS`, instruções `INSERT` geram um erro a menos que você especifique valores explicitamete para todas as colunas que exigem um valor não-`NULL`. Veja Seção 2.3.3 [Opções do `configure`], Página 98.
- Você pode encontrar o valor usado por uma coluna `AUTO_INCREMENT` com a função `mysql_insert_id`. Veja Seção 12.1.3.32 [`mysql_insert_id()`], Página 803.

Se você utilizar instruções `INSERT ... SELECT` ou `INSERT ... VALUES` com lista de valores múltiplos, você pode utilizar a função `mysql_info()` da API C para obter informação sobre a consulta. O formato da string de informação é mostrado aqui:

```
Records: 100 Duplicates: 0 Warnings: 0
```

`Duplicates` indica o número de linhas que não puderam ser inseridas porque duplicariam alguns valores de índices únicos existentes. `Warnings` indica o número de tentativas de inserção de um valor em uma coluna que de alguma forma estava problematico. Avisos (Warnings) podem ocorrer sob qualquer uma das seguintes condições:

- Inserir `NULL` em uma coluna declarada com `NOT NULL`. A coluna é definida com o seu valor padrão.
- Definir uma coluna numérica com um valor que esteja fora da faixa permitida. O valor é revertido para final apropriado da faixa.
- Definir uma coluna numérica com um valor como `'10.34 a'`. O lixo no final é eliminado e a parte numérica restante é inserida. Se o valor não fizer sentido como um número, é atribuido `0` a coluna.
- Inserir uma string em uma coluna `CHAR`, `VARCHAR`, `TEXT`, ou `BLOB` e que exceda o tamanho máximo da coluna. O valor é truncado para o tamanho máximo da coluna.
- Inserir um valor em uma coluna date ou time e que seja inválido para o tipo da coluna. A coluna é preenchida com o valor de zero apropriado para o tipo.

6.4.3.1 Sintaxe `INSERT ... SELECT`

```
INSERT [LOW_PRIORITY] [IGNORE] [INTO] nome_tabela [(column list)] SELECT ...
```

Com a instrução `INSERT ... SELECT` você pode inserir muitas linhas rapidamente em uma tabela a partir de outras tabelas

```
INSERT INTO tblTemp2 (fldID) SELECT tblTemp1.fldOrder_ID FROM tblTemp1 WHERE
tblTemp1.fldOrder_ID > 100;
```

As seguintes condições servem para uma instrução `INSERT ... SELECT`:

- Antes do MySQL 4.0.1, `INSERT ... SELECT` operava implicitamente em modo `IGNORE`. A partir do MySQL 4.0.1, você deve especificar `IGNORE` explicitamente para ignorar registros que causaria violação de chave duplicada.
- Antes do MySQL 4.0.14, a tabela alvo da instrução `INSERT` não pode aparecer na cláusula `FROM` da parte `SELECT` da consulta. Esta limitação é deixada na versão 4.0.14.
- Colunas `AUTO_INCREMENT` funcionam da mesma forma.
- Em programas C, Você pode usar a função `mysql_info()` da API C para obter informação sobre a consulta. Veja Seção 6.4.3 [INSERT], Página 579.
- Para assegurar que o log binário possa ser usado para re-criar a tabela original, MySQL não permitirá inserções concorrentes em um `INSERT ... SELECT`.

Você também pode utilizar `REPLACE` em vez de `INSERT` para sobrescrever linhas antigas. `REPLACE` é a contra parte para `INSERT IGNORE` no tratamento de novas linhas contendo valores de chave únicos que duplicam linhas antigas: As novas linhas são usadas para substituir as linhas antigas em vez de descartá-las.

6.4.3.2 Sintaxe `INSERT DELAYED`

```
INSERT DELAYED ...
```

A opção `DELAYED` para a instrução `INSERT` é um opção específica do MySQL que é muito útil se você tiver clientes que não possam esperar que o `INSERT` se complete. Este é um problema comum quando você utiliza o MySQL para fazer log e você também execute periodicamente instruções `SELECT` e `UPDATE` que levem muito tempo para completar. `DELAYED` foi intriduzido no MySQL versão 3.22.15. Ela é uma extensão do MySQL ao SQL-92.

`INSERT DELAYED` só funciona com tabelas `ISAM` e `MyISAM`. Note que como tabelas `MyISAM` suportam `SELECT` e `INSERT` concorrentes, se não houver blocos livres no meio do arquivo de dados, você raramente precisará utilizar `INSERT DELAYED` com `MyISAM`. Veja Seção 7.1 [MyISAM], Página 631.

Quando você utiliza `INSERT DELAYED`, o cliente irá obter um OK de uma vez e a linha será inserida quando a tabela não estiver sendo usada por outra thread.

Outro grande benefício do uso de `INSERT DELAYED` e que inserções de muitos clientes são empacotados juntos e escritos em um bloco. Isto é muito mais rápido que se fazer muitas inserções seperadas.

Note que atualmente as linhas enfileirdas só são armazenadas em memória até que elas sejam inseridas na tabela. Isto significa que se você matar o `mysqld` com `kill -9` ou se o `mysqld` finalizar inesperadamente, as linhas enfileiradas que não forma escritas em disco são perdidas.

A seguir temos uma descrição em detalhes do que acontece quando você utiliza a opção `DELAYED` com `INSERT` ou `REPLACE`. Nesta descrição, a “thread” e a thread que recebe um

comando `INSERT DELAYED` e "handler" é a thread que trata todas as instruções `INSERT DELAYED` de uma tabela particular.

- Quando uma thread executa uma instrução `DELAYED` em uma tabela, uma thread handler é criada para processar todas as instruções `DELAYED` para a tabela, se tal handler ainda não existir.
- A thread verifica se o handler já adquiriu uma trava `DELAYED`; se não, ele diz a thread handler para fazê-lo. A trava `DELAYED` pode ser obtida mesmo se outras threads tiver uma trava de `LEITURA` ou `ESCRITA` na tabela. De qualquer forma, o handler irá esperar por todas as travas `ALTER TABLE` ou `FLUSH TABLES` para se assegurar que a estrutura da tabela está atualizada.
- A thread executa a instrução `INSERT`, mas em vez de escrever a linha na tabela, ela põe uma cópia da linha final na fila que é gerenciada pela thread handler. Quaisquer erros de sintaxe são notificados pela thread e relatadas ao programa cliente.
- O cliente não pode relatar o número de duplicatas ou o valor `AUTO_INCREMENT` para a linha resultante; ele não pode obtê-los do servidor, pois o `INSERT` retorna antes da operação de inserção ser completada. Se você utiliza a API C, a função `mysql_info()` não irá retornar nada significante, pela mesma razão.
- O log binário é atualizado pela thread handler quando a linha é inserida na tabela. No caso de inserção de múltiplas linhas, o log binário é atualizado quando a primeira linha é inserida.
- Depois que todas as linhas `delayed_insert_limit` são escrita, o handle verifica se alguma instrução `SELECT` está pendente. Se estiver, ele permite que ela seja executada antes de continuar.
- Quando o handler não tiver mais linhas na fila, a tabela é destravada. Se nunhum comando `INSERT DELAYED` novo é recebido dentro de `delayed_insert_timeout` segundos, o handler termina.
- Se mais que `delayed_queue_size` estão pendentes em uma fila handler específica, a thread requisitando `INSERT DELAYED` espera até que haja espaçõ na fila. Isto é feito para assegurar que o servidor `mysqld` não utilize toda a memória área de memória de atraso.
- A thread handler irá aparecer na lista de processos do MySQL process list com `delayed_insert` na coluna `Command`. Ela será finalizada se você executar um comando `FLUSH TABLES` ou matá-la com `KILL thread_id`. No entanto, primeiro ela armazenará todas as linhas enfileiradas na tabela antes de sair. Durante este tempo ela não aceitará nenhum comando `INSERT` novo da outra thread. Se você executar um comando `INSERT DELAYED` depois disto, uma nova thread handler será criada.

  Note que o mostrado acima significa que o comando `INSERT DELAYED` tem prioridade maior que um comando `INSERT` normal se já houver um handler `INSERT DELAYED` em execução! Outro comando de atualização terá que esperar até que a fila `INSERT DELAYED` esteja vazia, alguém finalize a thread handler (com `KILL thread_id`), ou alguém execute `FLUSH TABLES`.
- As seguintes variáveis de estado fornecem informção sobre comandos `INSERT DELAYED`:

Variável	Significado
`Delayed_insert_threads`	Número de threads handler

`Delayed_writes`	Números de linhas escrita com `INSERT DELAYED`
`Not_flushed_delayed_rows`	Número de linhas esperando para serem escritas

Você pode visualizar estas variáveis com a instrução `SHOW STATUS` ou executando um comando `mysqladmin extended-status`.

Note que `INSERT DELAYED` é mais lento que um INSERT normal se a tabela não estiver em uso. Também há uma sobrecarga adicional para o servidor tratar um thread separada para cada tabela na qual você utiliza `INSERT DELAYED`. Isto significa que você só deve usar `INSERT DELAYED` quando você estiver certo de necessita dele!

6.4.4 Sintaxe `UPDATE`

```
UPDATE [LOW_PRIORITY] [IGNORE] nome_tabela
    SET nome_coluna1=expr1 [, nome_coluna2=expr2 ...]
    [WHERE definição_where]
    [ORDER BY ...]
    [LIMIT row_count]

ou

UPDATE [LOW_PRIORITY] [IGNORE] nome_tabela [, nome_tabela ...]
    SET nome_coluna1=expr1 [, nome_coluna2=expr2 ...]
    [WHERE definição_where]
```

`UPDATE` atualiza uma coluna em registros de tabelas existentes com novos valores. A cláusula `SET` indica quais colunas modificar e os valores que devem ser dados. A cláusula `WHERE`i, se dada, especifica quais linhas devem ser atualizadas. Senão todas as linhas são atualizadas. Se a cláusula `ORDER BY` é especificada, as linhas serão atualizada na ordem especificada.

Se você especificar a palavra-chave `LOW_PRIORITY`, a execução de `UPDATE` e atrasada até que nenhum outro cliente esteja lendo da tabela.

Se você especificar a palavra-chave `IGNORE`, a instrução não será abortada memso se nós obtermos erros de chaves duplicadas durante a atualização. Linhas que causem conflitos não serão atualizadas.

Se você acessa um coluna de `nome_tabela` em uma expressão, `UPDATE` utiliza o valor atual da coluna. Por exemplo, a seguinte instrução define a coluna `age` com o valor atual mais um:

```
mysql> UPDATE persondata SET age=age+1;
```

Atribuiçãoes `UPDATE` são avaliadas da esquerda para a direitat. Por exemplo, a seguinte instrução dobra a coluna `age` e então a incrementa:

```
mysql> UPDATE persondata SET age=age*2, age=age+1;
```

Se você define uma coluna ao valor que ela possui atualmente, o MySQL notará isto é não irá atualizá-la.

`UPDATE` retorna o número de linhas que forma realmente alteradas. No MySQL Versão 3.22 ou posterior, a função `mysql_info()` da API C retorna o número de linhas encontradas e atualizadas e o número de avisos que ocorreram durante o `UPDATE`.

A partir do MySQL versão 3.23, você pode utilizar `LIMIT row_count` para restringir o escopo do `UPDATE`. Uma cláusula `LIMIT` funciona da seguinte forma:

- Antes do MySQL 4.0.13, `LIMIT` é uma restrição que afeta as linhas. A instrução para assim que altera `row_count` linhas que satisfaçam a cláusula `WHERE`.
- Da versão 4.0.13 em diante, `LIMIT` é uma restrição de linhas correspondentes. A instrução para assim que ela encontrar `row_count` linhas que satisfaçam a cláusula `WHERE`, tendo elas sido alteradas ou não.

Se uma cláusula `ORDER BY` é utilizada (disponível no MySQL 4.0.0), as linhas serão atualizadas nesta ordem. Isto só é util em conjunto com `LIMIT`.

A partir da MySQL Versão 4.0.4, você também pode realizar operações `UPDATE` que cobrem múltiplas tabelas:

```
UPDATE items,month SET items.price=month.price
WHERE items.id=month.id;
```

O exemplo mostra um inner join usando o operador de vírgula, mas instruções `UPDATE` multi-tabelas podem usar qualquer tipo de join permitida na instrução `SELECT`, como `LEFT JOIN`.

Nota: você não pode utilizar `ORDER BY` ou `LIMIT` com multi-tabelas `UPDATE`.

6.4.5 Sintaxe DELETE

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM table_name
       [WHERE definição_where]
       [ORDER BY ...]
       [LIMIT row_count]

ou

DELETE [LOW_PRIORITY] [QUICK] [IGNORE] table_name[.*] [, table_name[.*] ...]
       FROM tabelas-referentes
       [WHERE definição_where]

ou

DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
       FROM nome_tabela[.*] [, nome_tabela[.*] ...]
       USING tabelas-referentes
       [WHERE definição_where]
```

`DELETE` deleta linhas de `nome_tabela` que satisfaçam a condição dada por `definição_where`, e retorna o número de registros deletados.

Se você exeecutar um `DELETE` sem cláusula `WHERE`, todas as linhas são deletadas. Se você o fizer no modo `AUTOCOMMIT`, isto irá funcionar como `TRUNCATE`. Veja Seção 6.4.6 [TRUNCATE], Página 587. No MySQL 3.23, `DELETE` sem uma cláusula `WHERE` retornará zero como o número de registros afetados.

Se você realmente quiser saber quantos registros são deletados quando você deletar todas as linhas mesmo sofrendo uma com a queda da velocidade, você pode utilizar uma instrução `DELETE` desta forma:

```
mysql> DELETE FROM nome_tabela WHERE 1>0;
```

Note que isto é muito mais lento que `DELETE FROM nome_tabela` sem cláusula `WHERE`, pois ele deleta uma linha de cada vez.

Se você especificar a palavra-chave `LOW_PRIORITY`, a execução do `DELETE` é atrasda até que nenhum outro cliente esteja lendo da tabela.

Para tabelas `MyISAM`, Se você especificar a palavra `QUICK`, o mecanismo de armazenamento não irá fundir os índices excluídos durante a deleção, o que pode aumentar a velocidade de certos tipos de deleção.

A velocidade das operações de deleção também pode ser afetadas pelos fatores discutidos em Seção 5.2.12 [Delete speed], Página 440.

A opção `IGNORE` faz com que o MySQL ignore todos os erros durente o processo de deleção dos registros. Erros encontrados durante o estágio de análise são processados da maneira comum. Erros que são ignorados devido ao uso desta opção são retornados como aviso. Esta opção aparece pela primeira vez na versão 4.1.1.

Em tabelas `MyISAM`, registros deletados são mantidos em uma lista encadeada e operções `INSERT` subsequentes reutilizam posições de registros antigos. Para recuperar espeços não utilizados e reduzir o tamanho do arquivo, utilize a instrução `OPTIMIZE TABLE` ou o utiliztário `myisamchk` para reorganizar as tabelas. `OPTIMIZE TABLE` é mais fácil, mas `myisamchk` é mais rápido. Veja Seção 4.6.1 [`OPTIMIZE TABLE`], Página 298 e Seção 4.5.6.10 [Optimization], Página 291.

O primeiro formato de delção de multi-tabelas é suportado a partir do MySQL 4.0.0. O segundo formato de deleção multi-tabelas é suportado a partir do MySQL 4.0.2.

A idéia é que apenas linhas coincidentes da tabelas listadas **antes** de `FROM` ou antes da cláusula `USING` são deletadas. O efeito é que você pode deletar l;inhas de muitas tabelas ao mesmo tempo e também ter tabelas adicionais que são utilizadas para busca.

O `.*` depois do nome da tabela existe apenas para ser compatível com o `Access`:

```
DELETE t1,t2 FROM t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id

ou

DELETE FROM t1,t2 USING t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id
```

No cso acima nós deletamos linhas coincidente apenas na tabela `t1` e `t2`.

O exemplo mostra um inner join usando o operador de vírgula, mas instruções `UPDATE` multi-tabelas podem usar qualquer tipo de join permitida na instrução `SELECT`, como `LEFT JOIN`.

Se uma cláusula `ORDER BY` é utilizada (disponível no MySQL 4.0.0), as linhas serão deletadas naquela ordem. Isto só é útil se usado em conjunto com `LIMIT`. Por exemplo:

```
DELETE FROM somelog
WHERE user = 'jcole'
ORDER BY timestamp
```

```
LIMIT 1
```

Isto irá deletar as entradas antigas (por `timestamp`) onde as linhas casam com a cláusula `WHERE`.

A opção específica do MySQL `LIMIT row_count` para `DELETE` diz ao servidor o número máximo de linhas a serem deletadas antes do controle retornar ao cliente. Isto pode ser usado para assegurar que uma comando `DELETE` específico mão tomará muito tempo, Você pode simplesmente repetir o comando `DELETE` até que o número de linhas afetadas seja menor que o valor `LIMIT`.

No MySQL 4.0, você pode especificar múltiplas tabelas na instrução `DELETE` para deletar linhas de uma ou mais tabelas dependendo de uma condição particular em várias tabelas. No entanto você não pode utilizar `ORDER BY` ou `LIMIT` em uma multi-tabela `DELETE`.

6.4.6 Sintaxe `TRUNCATE`

```
TRUNCATE TABLE nome_tabela
```

Na versão 3.23 `TRUNCATE TABLE` é mapeada para `COMMIT; DELETE FROM table_name`. Veja Seção 6.4.5 [DELETE], Página 585.

`TRUNCATE TABLE` difere de `DELETE FROM ...` do seguinte modo:

- Operações truncate apagam e recriam a tabela, o que é muito mais rápido que deletar registros um a um.
- Operações truncate não são seguras a transação; você iráobter um erro se você tiver uma transação ativa ou ativar um travamento de tabela.
- O número de linhas apagadas não é retornado.
- Uma vez que o arquivo de definição '`nome_tabela.frm`' deja válido, a tabela pode ser recriada deta forma, mesmo se o arquivo de dados ou de índice estiver corrompido.

`TRUNCATE` é uma extensão Oracle SQL. Esta instrução foi adicionada no MySQL 3.23.28, embora da versão 3.23.28 a 3.23.32, a palavra chave `TABLE` deva ser omitida.

6.4.7 Sintaxe `REPLACE`

```
    REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] nome_tabela [(nome_coluna,...)]
        VALUES (expressão,...),(...),...
ou  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] nome_tabela [(nome_coluna,...)]
        SELECT ...
ou  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] nome_tabela
        SET nome_coluna=expressão, nome_coluna=expressão,...
```

`REPLACE` funciona exatamente como o `INSERT`, exceto que se um registro antigo na tabela tem o mesmo valor que um novo registro em um índice `UNIQUE` ou `PRIMARY KEY`, o registro antigo é deletado antes que o novo registro seja inserido. Veja Seção 6.4.3 [INSERT], Página 579.

Em outras palavras, você não pode acessar os valores do registro antigo em uma instrução `REPLACE`. Em algumas versões antigas do MySQL aparentemente você podia fazer isto, mas era um bug que já foi arrumado.

Par aestar apto a utilizar `REPLACE` você deve ter privilégios `INSERT` e `DELETE` para a tabela.

Quando você utilizar um comando `REPLACE`, `mysql_affected_rows()` retornará 2 se a nova linha substituir uma linha antiga. Isto é porque uma linha foi inserida depois que a linha duplicada foi deletada.

Este fato torna fácil determinar se `REPLACE` adicionou ou subsitituiu uma linha: verifique se o valor de linhas afetadas é 1 (adicionado) ou 2 (substituido).

Note que a menos que a tabela utilize índices `UNIQUE` ou `PRIMARY KEY`, utilizar um comando `REPLACE` replace não faz sentido. Ele se torna equivalente a um `INSERT`, porque não existe índice a ser usado para determinar se uma nova linha duplica outra.

Seqgue aqui o algoritmo usado em mais detalhes: (Ele também é usado com `LOAD DATA ... REPLACE`.

```
- Insere a linha na tabela
  - Enquanto ocorrer erro de chave duplicada para chaves primária ou única
    - Reverte as chaves alteradas
    - Le as linha conflitantes da tabela através do valor da chave duplicada
    - Deleta as linhas conflitantes
    - Tenta inserir o chave primária e única original na árvore
```

6.4.8 Sintaxe `LOAD DATA INFILE`

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name.txt'
    [REPLACE | IGNORE]
    INTO TABLE nome_tabela
    [FIELDS
        [TERMINATED BY '\t']
        [[OPTIONALLY] ENCLOSED BY '']
        [ESCAPED BY '\\' ]
    ]
        [LINES
          [STARTING BY '']
        [TERMINATED BY '\n']
    ]
    [IGNORE número LINES]
    [(nome_coluna,...)]
```

A instrução `LOAD DATA INFILE` lê linhas de uma arquivo texto para uma tabela em uma velocidade muito alta. Se a palavra-chave `LOCAL` é especificada, ela é interpretada com respeito ao fim da conexão do cliente. Quando `LOCAL` é especificado, o arquivo é lido pelo programa cliente na máquina cliente e enviada ao servidor. Se `LOCAL` não é especificada, o arquivo deve estar localizado na máquina servidora e é lida diretamente pelo servidor (`LOCAL` está disponível no MySQL Versão 3.22.6 ou posterior).

Por razões de segurança, ao ler arquivos textos no servidor, os arquivos devem também estar no diretório de banco de dados ou serem lidos por todos. Também, para utilizar `LOAD`

`DATA INFILE` em arquivos do servidor, você deve ter privilégio `FILE` na máquina servidora. Veja Seção 4.3.7 [Privilégios fornecidos], Página 236.

A partir do MySQL 3.23.49 e MySQL 4.0.2 (4.0.13 no Windows) `LOCAL` só funcionará se o seu servidor e o seu cliente forem habilitados para permitir isto. Por exemplo so o `mysqld` foi iniciado com `--local-infile=0`, `LOCAL` não irá funcionar. Veja Seção 4.3.4 [LOAD DATA LOCAL], Página 231.

Se você especificar a palavra-chave `LOW_PRIORITY`, a execução da instrução `LOAD DATA` é atrasada até nenhum outro cliente estar lendo a tabela.

Se você especificar a palavra-chave `CONCURRENT` com uma tabela `MyISAM`, outras threads podem retornar dados da tabela enquanto `LOAD DATA` está executando. Utilizar esta opção irá afetar o desempenho de `LOAD DATA` um pouco, mesmo se nenhuma outra thread utilizar a tabela ao mesmo tempo.

Utilizar `LOCAL` será um pouco mais lento que deixar o servidor acessar os arquivos diretamente, pois o conteúdo do arquivo deve ser enviado pela conexão da máquina cliente até a máquina servidora. Por outro lado, você não precisa de ter o privilégio `FILE` para carregar arquivos locais.

Se você estiver utilizando uma versão do MySQL anterior a 3.23.24, você não poderá ler de um FIFO com `LOAD DATA INFILE`. Se você precisar ler de um FIFO (por exemplo a saída de gunzip), utilize `LOAD DATA LOCAL INFILE`.

Você também pode carregar arquivo de dados utilizado o utilitário `mysqlimport`; ele opera enviando um comando `LOAD DATA INFILE` para o servidor. A opção `--local` faz com que `mysqlimport` leia ao arquivo de dados a partir da máquina cliente. Você pode especificar a opção `--compress` para conseguir melhor desempenho sobre redes lentas se o cliente e o servidor suportar protocolos compactados.

Ao localizar arquivos na máquina servidora, o servidor utiliza as segintes regras:

- Se um caminho absoluto é dado, o servidor utiliza o caminho desta forma.
- Se um caminho relativo com um ou mais componentes é dados, o servidor busca o arquivo em relação ao diretório de dados do servidor.
- Se um nome de arquivo sem nenhum componente é dado, o servidor procura pelo arquivo no diretorio de banco de dados do banco de dados atual.

Note que estas regras significam que um arquivo chamado '`./myfile.txt`' é lido no diretório de dados do servidor, enquanto um arquivo chamado '`myfile.txt`' lê o diretório de dados do naco de dados atual. Por exemplo, a seguinte instrução `LOAD DATA` lê o arquivo '`data.txt`' do diretório de dados de `db1` pois `db1` é o banco de dados atual, mesmo que a instrução carrega explicitamente o arquivo em uma tabela no banco de dados `db2`:

```
mysql> USE db1;
mysql> LOAD DATA INFILE "data.txt" INTO TABLE db2.my_table;
```

As palavras-chave `REPLACE` e `IGNORE` controlam o tratamento de entrada de registros que duplicam linhas existentes em valores de chave única.

Se você especificar `REPLACE`, as linhas inseridas substituirão as linhas existentes (em outras palavras, linhas que tiverem o mesmo valor de um índice primário ou único como linhas existentes). Veja Seção 6.4.7 [REPLACE], Página 587.

Se você especificar `IGNORE`, registros inseridos que duplicam uma linha existente em um valor de chave única será ignorados. Se você não especificar nenhuma das opções, o comportamento depende de se a palavra chave `LOCAL` é especificada ou não. Sem `LOCAL`, um erro ocorre quando um valor de chave duplicada é encontrado, e o resto do arquivo texto é ignorado. Com `LOCAL` o comportamento padrão é o mesmo de quando `IGNORE` for especificado, isto é porque o servidor não tem como parar no meio da operação.

Se você quiser ignorar as restrições de chaves estrangeiras durante a carga você pode faze `SET FOREIGN_KEY_CHECKS=0` antes de executar `LOAD DATA`.

Se você utiliza `LOAD DATA INFILE` em uma tabela `MyISAM` vazia, todos os índices não-únicos são criados em um batch separado (como em `REPAIR`). Isto normalmente torna `LOAD DATA INFILE` muito mais rápido quando você tem diversos índices. Normalmente isto é muito rápido mas em casos extremos você pode tornar o índice mais rápido ainda desligando-os com `ALTER TABLE .. DISABLE KEYS` e usando `ALTER TABLE .. ENABLE KEYS` para recriar os índices. Veja Seção 4.5.6 [Manutenção de tabelas], Página 280.

`LOAD DATA INFILE` é o complemento de `SELECT ... INTO OUTFILE`. Veja Seção 6.4.1 [SELECT], Página 562. Para gravar dados de uma tabela em um arquivo, use `SELECT ... INTO OUTFILE`. Para ler o arquivo de volta em uma tabela, use `LOAD DATA INFILE`. A sintaxe das cláusulas `FIELDS` e `LINES` é a mesma para ambos os comandos. Ambas as cláusulas são opicionais, mas `FIELDS` deve preceder `LINES` se ambos são especificados.

Se você especificar uma cláusula `FIELDS`, cada uma das subcláusulas (`TERMINATED BY`, `[OPTIONALLY] ENCLOSED BY`, e `ESCAPED BY`) também são opicionais, exceto pelo fato de que você deve especificar pelo menos uma delas.

Se você não especificar uma cláusula `FIELDS`, o padrão é o mesmo que se você tivesse escrito isto:

```
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
```

Se você não especificar uma cláusula `LINES`, o padrão é o mesmo que se você tivesse escrito isto:

```
LINES TERMINATED BY '\n'
```

Nota: Se você gerou o arquivo texto no Windows, você deve alterar o mostrado acima para: `LINES TERMINATED BY '\r\n'` já que o Windows utiliza dois caracteres como um terminador de linha. Alguns programas como `wordpad`, pode usar `\r` como terminador de linha.

Se todas as linas que você deseja ler tem um prefixo comum que você quer saltar, você pode usar `LINES STARTING BY prefix_string`.

Em outras palavras, o padrão faz com que `LOAD DATA INFILE` funcione da seguinte maneira ao se ler uma entrada:

- Procure pelo limite da linha em linhas novas.
- Se `LINES STARTING BY prefix` for usado, lê até que o prefixo seja encontrado e começa a ler o caracter seguinte ao prefixo. Se a linha não inclui o prefico e;a será saltada.
- Quebre a linha em campos na tabulações.
- Não espere que os campos estejam entre aspas.
- Interprete a ocorrência de tabulações, novas linhas ou '\' precedidos por '\' como caracteres literias que são parte dos valores dos campos.

Inversamente, os padrões fazem `SELECT ... INTO OUTFILE` funcionar da seguinte forma ao escrever as saídas:

- Escreva tabulações entre os campos.
- Não coloque campos entre aspas.
- Utilize '\' para considerar como parte dos campos instâncias de tabulação, nova linha ou '\' que estejam dentro dos valores dos campos.
- Escreva novas linhas no fim de cada linha.

Note que para escrever `FIELDS ESCAPED BY '\\'`, você deve especificar duas barras invertidas para que o valor seja lido como uma única barra invertida.

A opção `IGNORE número LINES` pode ser utilizado para ignorar linhas no inicio do arquivo. Por exemplo, você pode usar `IGNORE 1 LINES` para saltar uma linha de cabeçalho contendo nomes de colunas:

```
mysql> LOAD DATA INFILE "/tmp/file_name" INTO TABLE test IGNORE 1 LINES;
```

Quando você utiliza `SELECT ... INTO OUTFILE` em conjunto com `LOAD DATA INFILE` para escrever os dados de um banco de dados em um arquivo e então ler o arquivo de volta no banco de dados posteriormente, as opções para tratamento de linhas e campos para ambos os comandos devem coincidir. Senão, `LOAD DATA INFILE`não irá interpretar o conteúdo do arquivo de forma apropriada. Suponha que você utilize `SELECT ... INTO OUTFILE` para escrever um arquivo com os campos separados por vírgulas:

```
mysql> SELECT * INTO OUTFILE 'data.txt'
    ->          FIELDS TERMINATED BY ','
    ->          FROM ...;
```

Para ler o arquivo delimitado com vírgula de volta, a instrução correta seria:

```
mysql> LOAD DATA INFILE 'data.txt' INTO TABLE table2
    ->           FIELDS TERMINATED BY ',';
```

Se você tentasse ler do arquivo com a instrução abaixo, não iria funcionar pois ela instrui `LOAD DATA INFILE` a procurar por tabulações entre campos:

```
mysql> LOAD DATA INFILE 'data.txt' INTO TABLE table2
    ->           FIELDS TERMINATED BY '\t';
```

O resultado desejado é que cada linha de entrada fosse interpretada como um único campo.

`LOAD DATA INFILE` pode ser usado para ler arquivos obtidos de fontes externas. Por exemplo, um arquivo no formato dBASE terá campos separados por vírgulas e entre aspas duplas. Se as linhas no arquivo são terminadas por com uma nova linha, o comando mostardo aqui ilustra as opções do tratamento de campos e linhas que você usaria pra carregar o arquivo. the file:

```
mysql> LOAD DATA INFILE 'data.txt' INTO TABLE nome_tabela
    ->           FIELDS TERMINATED BY ',' ENCLOSED BY '"'
    ->           LINES TERMINATED BY '\n';
```

Qualquer uma das opções de tratamento de campos e linhas podem especificar uma string vazia (''). Se não for vazio, os valores de `FIELDS [OPTIONALLY] ENCLOSED BY` e `FIELDS ESCAPED BY` devem ser um caracter simples. Os valores de `FIELDS TERMINATED BY` e `LINES`

TERMINATED BY podem ser mais de uma caracter. Por exemplo, para escrever linhas terminadas pelos par retorno de carro/alimentação de linha, ou para ler um arquivo contendo tais linhas, especifique uma cláusula LINES TERMINATED BY '\r\n'.

Por exemplo, para ler um arquivo de piadas, que são separadas com uma linha de %%, em uma tabela SQL, você pode fazer:

```
CREATE TABLE jokes (a INT NOT NULL AUTO_INCREMENT PRIMARY KEY, joke TEXT
NOT NULL);
LOAD DATA INFILE "/tmp/jokes.txt" INTO TABLE jokes FIELDS TERMINATED BY ""
LINES TERMINATED BY "\n%%\n" (joke);
```

FIELDS [OPTIONALLY] ENCLOSED BY controla a citação dos campos. Para saida (SELECT ... INTO OUTFILE), se você omitir a palavra OPTIONALLY, todos os campos estarão entra o caracter ENCLOSED BY. Um exemplo de tal saída (usando vírgula como delimitador de campo) é mostrado abaixo:

```
"1","a string","100.20"
"2","a string containing a , comma","102.20"
"3","a string containing a \" quote","102.20"
"4","a string containing a \", quote and comma","102.20"
```

Se você especificar OPTIONALLY, o caracter ENCLOSED BY só é usados para delimitar campos CHAR e VARCHAR:

```
1,"a string",100.20
2,"a string containing a , comma",102.20
3,"a string containing a \" quote",102.20
4,"a string containing a \", quote and comma",102.20
```

Note que a ocorrência de caracter ENCLOSED BY dentro do valor do campo é indicado colocando um caracter ESCAPED BY antes dele. Note também que se você especificar um valor ESCAPED BY vazio, é possível gerar saídas que não poderão ser lidas aprorpiadamente por LOAD DATA INFILE. Por exemplo, a saída mostrada seria apareceria como a seguir se o caracter de escape fosse vazio. Observe que o segundo campo na quarta linha contém uma vírgula seguida de aspas, o que (erroneamente) parece terminar o campo:

```
1,"a string",100.20
2,"a string containing a , comma",102.20
3,"a string containing a " quote",102.20
4,"a string containing a ", quote and comma",102.20
```

Para entrada, o caracter ENCLOSED BY, se presente, será eliminado do fim dos valores dos campos. (Isto é verdade se OPTIONALLY for especificado; OPTIONALLY não tem efeito na interpretação da entrada). A ocorrência de caracteres ENCLOSED BY precedido pelo caracter ESCAPED BY são interpretados como parte do campo atual.

Se o campo começa com o caracter ENCLOSED BY, instâncias daquele caracter são reconhecidos como terminação de um valor do campo apenas se seguido pelo campo ou sequência de linah TERMINATED BY. Para evitar ambiguidade, ocorrências do caracter ENCLOSED BY dentro de um valor de campo pode ser duplicado e será interpretado como uma única instância do caracter. Por exemplo, se ENCLOSED BY '"' for especificado, aspas serão tratadas como mostrado abaixo:

```
"The ""BIG"" boss"  -> The "BIG" boss
```

```
The "BIG" boss      -> The "BIG" boss
The ""BIG"" boss    -> The ""BIG"" boss
```

`FIELDS ESCAPED BY` controla como escrever ou ler caracteres especiais. Se o caracter `FIELDS ESCAPED BY` não estivaer vazio, ele será usado para preceder o seguinte caracter de saída:

- O caracter `FIELDS ESCAPED BY`
- O caracter `FIELDS [OPTIONALLY] ENCLOSED BY`
- O primeiro caracter dos valores `FIELDS TERMINATED BY` e `LINES TERMINATED BY`
- ASCII `0` (o que é escrito seguido de um caracter de escape é ASCII `'0'`, não o byte de valor zero).

Se o caracter `FIELDS ESCAPED BY` estiver vazio, nenhum caracter será "escapado". Provavelmente não é uma boa idéia especificar um caracter de escape vazio, principalmente se os valores dos campos em seus conter qualquer caracter na lista dada.

Para entradas, se o caracter `FIELDS ESCAPED BY` não estiver vazio, as ocorrências daquele caracter são eliminadas e o caracter seguinte é tomado como parte do valor do campo. As exceções são um '`0`' ou '`N`' "escapado" (por exemplo, `\0` ou `\N` se o caracter de escape for '`\`'). Estas sequencias são interpretadas como os ASCII `0` (um byte de valor zero) e `NULL`. Veja abaixo as regras no tratamento de `NULL`.

Para maiores informações sobre a sintaxe '\'-escape, veja Seção 6.1.1 [Literals], Página 469.

Em certos casos, as opções de tratamento de campoe e linhas se interagem:

- Se `LINES TERMINATED BY` é uma string vazia e `FIELDS TERMINATED BY` não é vazio, as linhas também serão terminadas com `FIELDS TERMINATED BY`.
- Se os valores `FIELDS TERMINATED BY` e `FIELDS ENCLOSED BY` são ambos vazios (`''`), um formato de linha de tamanhos fixos (sem delimitadores) é utilizada. Com formato de linhas de tamanho fixo, nenhum deliitador é usado entre os campos (mas você ainda pode ter um terminador de linha). Valores de colunas são escritos e lidos usando o tamanho definido das colunas. Por exemplo, se uma coluna é declarada como `INT(7)`, os valores das colunas são escritos utilizando campos de 7 caracteres. Na saída, os valores das colunas são obtidos lendo 7 caracteres.

  `LINES TERMINATED BY` ainda é usado para separar linhas. Se uma linha não contém todos os campos, o resto dos campos serão configurados com o seu valor padrão. Se você não tiver um terminador de linha, você deve defini-lo com `''`. Neste caso o arquivo texto deve conter todos os campos para cada linha.

  O formato de linhas de tamanho fixo também afetam o tratamento de valores `NULL`; veja abixo. Note que este formato não funciona se você estiver utilizando um conjunto de caracteres mulyi-byte.

O tratamento do valor `NULL` varia, dependendo das opções de `FIELDS` e `LINES` que voce usar:

- Para os valores `FIELDS` e `LINES` padrões, `NULL` é escrito como `\N` para saída e `\N` é lido como `NULL` para as entradas (assumindo que o caracter `ESCAPED BY` é '`\`').
- Se `FIELDS ENCLOSED BY` não for vazio, um campo contendo a palavra literal `NULL` como seu valor é lido como um valor `NULL` (isto difere da palavra `NULL` entre os caracteres `FIELDS ENCLOSED BY`, a qual é lida como a string `'NULL'`).
- Se `FIELDS ESCAPED BY` for vazio, `NULL` é escrito como a palavra `NULL`.

- Com os formatos de tamanho fixos (que acontecem quando `FIELDS TERMINATED BY` e `FIELDS ENCLOSED BY` estiverem ambos vazios), `NULL` é escrito como uma string vazia. Note que isto faz com que os valores `NULL` e uma string vazia na tabela serão indistingüíveis quando escritas no arquivo pois elas são ambas escritas como strings vazias. Se você precisar estar saber diferenciar as duas ao ler o arquivo de volta, você não deve utilizar o formato de tamanho fixo.

Alguns casos não são suportados por `LOAD DATA INFILE`:

- Linhas de tamanho fixo (`FIELDS TERMINATED BY` e `FIELDS ENCLOSED BY` vazios) e colunas `BLOB` ou `TEXT`.
- Se você especificar um separador que é igual ao prefixo do outro, `LOAD DATA INFILE` não poderá interpretar a entratada apropriadamente. Por exemplo, a seguinte cláusula `FIELDS` causaria problemas:

```
FIELDS TERMINATED BY '"' ENCLOSED BY '"'
```

- Se `FIELDS ESCAPED BY` estiver vazio, um valor de campo que contém uma ocorrência de `FIELDS ENCLOSED BY` ou `LINES TERMINATED BY` seguido por valores `FIELDS TERMINATED BY` fará com que `LOAD DATA INFILE` pare de ler um campo ou linha antes do esperado. Isto ocorre porque `LOAD DATA INFILE` não pode determinar apropriadamente onde o valor de campo ou linha acaba.

A oseguinte exemplo carrega todas as colunas da tablea `persondata`:

```
mysql> LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata;
```

Nenhuma lista de campo é especificada, assim `LOAD DATA INFILE` espera linhas de entradas que contenha um campo para cada coluna da tabela. Os valores padrões de `FIELDS` e `LINES` são usados.

Se você deseja carregar somente algumas das colunas das tabelas, especifique uma lista de campos:

```
mysql> LOAD DATA INFILE 'persondata.txt'
    ->           INTO TABLE persondata (col1,col2,...);
```

Você deve especificar uma lista de campos se a ordem dos campos no arquivo de entrada diferem da ordem das colunas na tabela. Senão o MySQL não poderá dizer como combinar os campos da entrada nas colunas da tabela.

Se uma linha tiver poucos campos, as colunas para os quais o campo de entrada não estiverem presentes serão definidas com o valor padrão. Atribuição de valor padrão é descrito em Seção 6.5.3 [`CREATE TABLE`], Página 598.

Um valor de campo vazio é interpretado de forma diferente de que se o valor do campo estiiver faltando:

- Para tipos string, a coluna é definida com uma string vazia.
- Para tipos numéricos, a coluna é definida com `0`.
- Para tipos de data e hora, a coluna é definida com o valor "zero" apropriado para o tipo. Veja Seção 6.2.2 [Tipos date e time], Página 489.

Note que estes são os mesmos valores que resultam se você atribuir uma string vazia explicitamente a um tipo string, numérico, de data ou de hora em uma instrução `INSERT` ou `UPDATE`.

Colunas `TIMESTAMP` só são definidas com a hora e data atual se houver um valor `NULL` para a coluna (isto é, `\N`), ou (apenas para a primeira coluna `TIMESTAMP`) se a coluna `TIMESTAMP` esta a esquerda da lista de campos quando esta for especificada.

Se uma linha de entrada tiver muitos campos, os campos extras serão ignorados e o número de avisos é incrementado. Note que antes do MySQL 4.1.1 o aviso é apenas um número que indica que alguma coisa deu errado. No MySQL 4.1.1 você pode fazer `SHOW WARNINGS` para obter mais informações sobre o que deu errado.

`LOAD DATA INFILE` considera todas as entradas como strings, assim você não pode utiliar valores numéricos para colunas `ENUM` ou `SET` do mesmo modo que você pode com instruções `INSERT`. Todos os valores `ENUM` e `SET` devem ser especificados como strings!

Se você estiver usando a API C, você pode obter informações sobre a consulta chamando a função `mysql_info()` da API C quando a consulta `LOAD DATA INFILE` terminar. O formato da string de informação é mostrado aqui:

```
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

Avisos ocorrem sob as mesmas circuntâncias que quando são inseridos via instrução `INSERT` (veja Seção 6.4.3 [INSERT], Página 579), exceto que `LOAD DATA INFILE` também gera avisos quando houver poucos ou muitos campos na linha de entrada. Os avisos não são armazenados em nenhum local; o número de avisos só pode ser utilizado como uma indicação se tudo correr bem.

Se você obter avisos e quiser saber exatamente porque eles ocorreram, um modo de se fazer isto é utilizar `SELECT ... INTO OUTFILE` em outro arquivo e camporá-lo ao arquivo de entrada original.

Se você precisar que `LOAD DATA` leia de um pipe, você pode utilizar o seguinte truque:

```
mkfifo /mysql/db/x/x
chmod 666 /mysql/db/x/x
cat < /dev/tcp/10.1.1.12/4711 > /nt/mysql/db/x/x
mysql -e "LOAD DATA INFILE 'x' INTO TABLE x" x
```

Se você estiver usando uma versão do MySQL a anterior a 3.23.25 você só poderá fazer o descrito acima com `LOAD DATA LOCAL INFILE`.

No MySQL 4.1.1 você pode usar `SHOW WARNINGS` para conseguir a lista do primeiros `max_error_count` avisos. Veja Seção 4.6.8.9 [SHOW WARNINGS], Página 323.

Para mais informações sobre a eficiência de `INSERT` versus `LOAD DATA INFILE` e a melhora na velocidade de `LOAD DATA INFILE`, Veja Seção 5.2.10 [Velocidade da inserção], Página 438.

6.4.9 Sintaxe `HANDLER`

```
HANDLER nome_tabela OPEN [ AS alias ]
HANDLER nome_tabela READ nome_indice { = | >= | <= | < } (valor1,valor2,...)
    [ WHERE ... ] [LIMIT ... ]
HANDLER nome_tabela READ nome_indice { FIRST | NEXT | PREV | LAST }
    [ WHERE ... ] [LIMIT ... ]
HANDLER nome_tabela READ { FIRST | NEXT }
    [ WHERE ... ] [LIMIT ... ]
HANDLER nome_tabela CLOSE
```

A instrução `HANDLER` fornece acesso direto a interface do mecanismo de armazenamento de tabelas `MyISAM`.

A primeira forma da instrução `HANDLER` abre uma tabela, tornando a acessível através de subsequentes instruções `HANDLER ... READ`. Este objeto de tabela não é copartilhada com outras threads e não serão fechadas até que as chamadas de thread `HANDLER nome_tabela CLOSE` ou a thread termine.

A segunda forma busca um registro (ou mais, especificado pela cláusula `LIMIT`) onde o índice especificado satisfaz os valores dados e a condição `WHERE` é encontrada. Se você tiver um índice multi-coluna, especifique as colunas do índice como uma lista separadas por vírgulas. Especifique o valor de todas as colunas no índice, ou especifique valores para o prefixo mais a esquerda das colunas índices. Suponha que um índice inclui três colunas chamadas `col_a`, `col_b`, e `col_c`, nesta ordem. A instrução `HANDLER` pode especificar valores para todas as três colunas no índice, ou para as colunas no prefixo mais a esquerda. Por exemplo:

```
HANDLER ... index_name = (col_a_val,col_b_val,col_c_val) ...
HANDLER ... index_name = (col_a_val,col_b_val) ...
HANDLER ... index_name = (col_a_val) ...
```

A terceira forma busca uma linha (ou mais, especificado pela cláusula `LIMIT`) da tabela na ordem do índice, correspondendo a condição `WHERE`.

A quarta forma (sem especificação de índice) busca um registro (ou mais, especificado pela cláusula `LIMIT`) da tabela na ordem natural da linhas (como armazenado no arquivo de dados) de acordo com a condição `WHERE` é mais rápido que `HANDLER nome_tabela READ nome_indice` quando é necessária uma varredura completa da tabela.

`HANDLER ... CLOSE` fecha uma tabela que foi aberta com `HANDLER ... OPEN`.

Nota: Se você estiver utilizando a interface `HANDLER` para `PRIMARY KEY` você deve se lembrar de colocar o nome entre aspas: `HANDLER tbl READ ‘PRIMARY‘ > (...)`

`HANDLER` é uma instrução de baixo nível. Por exemplo, ela não fornece consitência. Isto é, `HANDLER ... OPEN` **NÃO** pega uma imagem instântanea da tabela, e **NÃO** trava a tabela. Isto significa que depois que um `HANDLER ... OPEN` é feito, os dados da tabela podem ser modificados (por esta ou outra thread) e estas modificações podem aparecer apenas parcialmente nas buscas `HANDLER ... NEXT` ou `HANDLER ... PREV`.

As razões para se utilizar esta interface em vez do SQL normal são:

- Ela é mais rápida que `SELECT` porque:
  - Um mecanismo de armazenamento designado é alocado pela thread em `HANDLER OPEN`.
  - Existe menos análise envolvida.
  - Nào existe sobrecaga de otimização e verificação de consultas.
  - A tabela utilizada não precisa estar travada em pedidos de dois handlers.
  - A interface handler não precisa fornecer uma aprência consistente dos dados (por exemplo, dirty-reads são permitidas), assim o mecanismo de armazenamento pode fazer otimizações que o SQL normalmente não permite.
- É muito mais fácil portar aplicações que usam interface como ISAM para o MySQL.
- Ele permite se fazer uma travessia em um banco de dados de uma maneira que não é facil (em alguns casos impossível) de fazer com SQL. A interface handler é um modo

mais natural de mostrar dados ao trabalhar com aplicações que fornecem uma interface interativa com o usuário para o banco de dados.

6.4.10 Sintaxe DO

```
DO expressão, [expressão, ...]
```

Executa a expressão mas não retorna nenhum resultado. Este é um modo curto de `SELECT expressão, expressão`, mas tem a vantagem de ser rápida quando você não se preocupa com o resultado.

Ele é útil principalmente com funções que tem efeitos em um dos lados, como `RELEASE_LOCK`.

6.5 Definição de Dados: CREATE, DROP e ALTER

6.5.1 Sintaxe CREATE DATABASE

```
CREATE DATABASE [IF NOT EXISTS] nome_bd
```

`CREATE DATABASE` cria um banco de dados com o nome dados.

As regras para os nomes de banco de daddos permitidos são daods em Seção 6.1.2 [Legal names], Página 472. Um erro ocorre se o banco de dados já existir e você não especificou `IF NOT EXISTS`.

Banco de dados no MySQL são implementados como diretórios contendo arquivos que correspondem a tabelas no banco de dados. Por não haver tabelas em um banco de dados quando ele é criado, a instrução `CREATE DATABASE` apenas cria um diretório sob o diretório de dados do MySQL.

Você também pode criar banco de dados com `mysqladmin`. Veja Seção 4.9 [Scripts do Lado do Servidor], Página 345.

6.5.2 Sintaxe DROP DATABASE

```
DROP DATABASE [IF EXISTS] nome_bd
```

`DROP DATABASE` deleta todos as tabelas no banco de dados e deleta o banco de dados. Se você fizer um `DROP DATABASE` em um banco de dados ligado simbolicamente, o link e o banco de dados original são deletados. **Tenha cuidado com este comando!**

`DROP DATABASE` retorna o número de arquivos que foram removidos do diretorio de banco de dados. Para tabelas `MyISAM`, isto é três vezes o número de tabelas, pois cada tabela corresponde a um arquivo '`.MYD`', um arquivo '`.MYI`' e um arquivo '`.frm`'.

O comando `DROP DATABASE` remove do diretório de banco de dados dado todos os arquivos com a seguinte extensão:

Ext	Ext	Ext	Ext
.BAK	.DAT	.HSH	.ISD
.ISM	.ISM	.MRG	.MYD
.MYI	.db	.frm	

Todos os subdiretórios que consistem de 2 digitos (diretórios `RAID`) também são removidos.

No MySQL Versão 3.22 ou posterior, você pode utilizar a palavra chave `IF EXISTS` para prevenir da ocorrência de um erro se o banco de dados não existir.

Você também pode deletar um banco de dados com `mysqladmin`. Veja Seção 4.9 [Scripts do Lado do Servidor], Página 345.

6.5.3 Sintaxe `CREATE TABLE`

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nome_tabela [(definição_create,...)]
[table_options] [select_statement]

ou

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nome_tabela [(]LIKE nome_antigo_tabela[)];

definição_create:
  nome_coluna tipo [NOT NULL | NULL] [DEFAULT valor_padrão] [AUTO_INCREMENT]
            [[PRIMARY] KEY] [COMMENT 'string'] [definição_referência]
  | [CONSTRAINT [symbol]] PRIMARY KEY (index_col_name,...)
  | KEY [nome_indice] (index_nome_coluna,...)
  | INDEX [nome_indice] (index_nome_coluna,...)
  | [CONSTRAINT [symbol]] UNIQUE [INDEX] [index_name] (index_col_name,...)
  | FULLTEXT [INDEX] [nome_indice] (index_nome_coluna,...)
  | [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (index_col_name,...)
            [definição_referência]
  |    CHECK (expr)

tipo:
        TINYINT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |    SMALLINT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |    MEDIUMINT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |    INT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |    INTEGER[(tamanho)] [UNSIGNED] [ZEROFILL]
  |    BIGINT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |    REAL[(tamanho,decimais)] [UNSIGNED] [ZEROFILL]
  |    DOUBLE[(tamanho,decimais)] [UNSIGNED] [ZEROFILL]
  |    FLOAT[(tamanho,decimais)] [UNSIGNED] [ZEROFILL]
  |    DECIMAL(tamanho,decimais) [UNSIGNED] [ZEROFILL]
  |    NUMERIC(tamanho,decimais) [UNSIGNED] [ZEROFILL]
  |    CHAR(tamanho) [BINARY | ASCII | UNICODE]
  |    VARCHAR(tamanho) [BINARY]
  |    DATE
  |    TIME
  |    TIMESTAMP
  |    DATETIME
  |    TINYBLOB
  |    BLOB
  |    MEDIUMBLOB
```

```
  |    LONGBLOB
  |    TINYTEXT
  |    TEXT
  |    MEDIUMTEXT
  |    LONGTEXT
  |    ENUM(value1,value2,value3,...)
  |    SET(value1,value2,value3,...)

index_nome_coluna:
        nome_coluna [(tamanho)] [ASC | DESC]

definição_referência:
        REFERENCES nome_tabela [(index_nome_coluna,...)]
                   [MATCH FULL | MATCH PARTIAL]
                   [ON DELETE opção_referência]
                   [ON UPDATE opção_referência]

opção_referência:
        RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

opções_tabela: table_option [table_option] ...

opções_tabela:
TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE | MRG_MYISAM | MYISAM }
   | AUTO_INCREMENT = #
   | AVG_ROW_LENGTH = #
   | CHECKSUM = {0 | 1}
   | COMMENT = 'string'
   | MAX_ROWS = #
   | MIN_ROWS = #
   | PACK_KEYS = {0 | 1 | DEFAULT}
   | PASSWORD = 'string'
   | DELAY_KEY_WRITE = {0 | 1}
   | ROW_FORMAT = { DEFAULT | DYNAMIC | FIXED | COMPRESSED }
   | RAID_TYPE = { 1 | STRIPED | RAID0 } RAID_CHUNKS=#  RAID_CHUNKSIZE=#
   | UNION = (table_name,[table_name...])
   | INSERT_METHOD = { NO | FIRST | LAST }
   | DATA DIRECTORY = 'caminho absluto para o diretório'
   | INDEX DIRECTORY = 'caminho absluto para o diretório'
   | DEFAULT CHARACTER SET character_set_name [COLLATE collation_name]

instrução_select:
[IGNORE | REPLACE] [AS] SELECT ...  (Alguma instrução válida)
```

`CREATE TABLE` cria uma tabela com op nome dado no banco de dados atual.

As regras para nomes de tabelas permitidos são dados em Seção 6.1.2 [Legal names], Página 472. Por padrão a tabela é criada no banco de dados atual. Um erro ocorre se não houver o banco de dados atual ou se a tabela já existir.

No MySQL Versão 3.22 ou posterior, o nome de tabela pode ser especificado como `nome_bd.nome_tabela` para criar a tabela em um banco de dados específico. Ele funciona sem se preoocupar se existe um banco de dados atual.

A partir do MySQL Versão 3.23, você pode usar a palavra-chave `TEMPORARY` qaundo você criar uma tabela. A tabela temporária é visível apenas a para a conexão atual, e será automaticamente deletada quando a conexão é fechada. Isto significa que duas conexões diferentes podem usar o mesmo nome de tabela temporária sem conflitos outras ou com uma tabela existente com o mesmo nome. (A tabela existente é escondida até que a tabela temporária seja deletada). A partir do MySQL 4.0.2 você deve ter o privilégio `CREATE TEMPORARY TABLES` para poder criar tabelas temporárias.

No MySQL Versão 3.23 ou posterior você pode utilizar as palavras-chaves `IF NOT EXISTS` para que não ocorra um erro se a tabela já existir. Note que não há verificação de que a tabela existente tem uma estrutura idêntica a aquela indicada pela instrução `CREATE TABLE`

A partir da versão 4.1.0, o atributo `SERIAL` pode ser usado com um alias para `BIGINT NOT NULL AUTO_INCREMENT UNIQUE`. Este é um recuros para compatibilidade.

Como no MySQL 3.23, você pode criar uma tabela de autra adicionando uma instrução `SELECT` no fim da instrução `CREATE TABLE`:

```
CREATE TABLE new_tbl SELECT * FROM orig_tbl;
```

Os índices não são transportados para a nova tabela, e algumas conversões de tipos de coluna podem ocorrer. Por exemplo, o atributo`AUTO_INCREMENT` não está preservado e colunas `VARCHAR` podem se tornar colunas `CHAR`.

Quando criar uma tabela com `CREATE ... SELECT`, de um apelido para qualquer chamada de função ou expressões em uma consulta. Se você não o fizer, a instrução `CREATE` pode falhar ou resultar em nomes de colunas indesejáveis.

```
CREATE TABLE artists_and_works
SELECT artist.name, COUNT(work.artist_id) AS number_of_works
FROM artist LEFT JOIN work ON artist.id = work.artist_id
GROUP BY artist.id;
```

No MySQL 4.1, você pode especificar explicitamente o tipo para uma coluna gerada:

```
CREATE TABLE foo (a tinyint not null) SELECT b+1 AS 'a' FROM bar;
```

No MySQL 4.1 você pode utilizar `LIKE` para criar uma tabela baseada em uma definição de outra tabela. No MySQL 4.1 você também pode especificar o tipo para uma coluna gerada:

```
CREATE TABLE new_tbl LIKE orig_tbl;
```

Cada tabela `nome_tabela` é representada por algum arquivo no diretório de banco de dados. No caso das tabelas tipo `MyISAM` você irá obter:

`CREATE TABLE ... LIKE` não copia nenhuma opção de tabela `DATA DIRECTORY` ou `INDEX DIRECTORY` que foi especificada para a tabela original.

Arquivo	**Proposito**
`nome_tabela.frm`	Arquivo de formato (definição) da tabela.

`nome_tabela.MYD` Arquivo de dados
`nome_tabela.MYI` Arquivo Índice

Para mais informações de propriedades de varios tipo de coluna, veja Seção 6.2 [Column types], Página 482:

- Se nem `NULL` nem `NOT NULL` for especificado, a coluna é tratada como se `NULL` fosse especificado.
- Uma coluna integer pode ter o atributo adicional `AUTO_INCREMENT`. Quando você insere um valor de `NULL` (recomendado) ou `0` em uma coluna `AUTO_INCREMENT` indexada, a coluna é definida com o valor da próxima sequência. Normalmente ele é `valor+1`, onde `valor` é o maior valor para a coluna column atualmente na tabela. A sequência de `AUTO_INCREMENT` começa com `1`. Veja Seção 12.1.3.32 [`mysql_insert_id()`], Página 803.

  A partir do MySQL 4.1.1, especificando o parâmetro `NO_AUTO_VALUE_ON_ZERO` para a opção do servidor `--sql-mode` ou a variável do servidor `sql_mode` permite que você aramzene `0` nas colunas `AUTO_INCREMENT` como `0`, em vez de gerar uma nova sequência de valores. Veja Seção 4.1.1 [Command-line options], Página 207.

  Se você deletar a linha contendo o valor máximo para uma coluna `AUTO_INCREMENT`, o valor será reutilizado por uma tabela `ISAM`, ou `BDB`, mas não por tabelas `MyISAM` ou `InnoDB`. Se você deletar todas as linhas na sua tabela com `DELETE FROM nome_tabela` (sem um `WHERE`) no modo `AUTOCOMMIT`, a sequencia será reiniciada em todos os tipos de tabela, exceto `InnoDB`. Veja Seção 7.5.12.5 [InnoDB auto-increment column], Página 675.

  Nota: Só pode haver uma coluna `AUTO_INCREMENT` por tabela, e ela deve ser indexada e não pode ter uma valor `DEFAULT`. No MySQL Versão 3.23, uma coluna `AUTO_INCREMENT` funcionará corretamente apenas se conter apenas valores positivos. Inserir um número negativo é considerado como a inserção de um número positivo muito grande. Isto ocorre para evitar problemaa de precisão quando os números vão de positivo para negativo e também para assegurar que não se obtenha, acidentalmente, uma coluna `AUTO_INCREMENT` que contenha 0.

  Em tabelas `MyISAM` e `BDB` você pode especificar colunas `AUTO_INCREMENT` secundárias em uma chave ulti-coluna. Veja Seção 3.6.9 [exemplo-AUTO_INCREMENT], Página 201.

  Para tornar MySQL compatível com alguns aplicativos ODBC, você pode encontrar o valor `AUTO_INCREMENT` da última linha inserida com a seguinte consulta:

  ```
  SELECT * FROM nome_tabela WHERE auto_col IS NULL
  ```

- Valores `NULL` são tratados em colunas `TIMESTAMP` de modo diferente de outros tipos de colunas. Você não pode armazenar um `NULL` literal em uma coluna `TIMESTAMP`; definindo a coluna com `NULL` lhe atribui a a data e a hora atual. Como colunas `TIMESTAMP` se comportam desta forma, os atributos `NULL` e `NOT NULL` não se aplicam de modo normal e são ignorados se você os especificar.

  Por outro lado, tornar o uso de colunas `TIMESTAMP` mais fácil para os clientes MySQL, o servidor relata que tal coluna pode ter o valor `NULL` atribuído (a que é verdade), mesmo que `TIMESTAMP` nunca contenham, realmente, um valor `NULL`. Você pode ver isto quando você utiliza `DESCRIBE nome_tabela` para obter informações sobre sua tabela.

  Note que definir uma coluna `TIMESTAMP` com `0` não é o mesmo que definí-la com `NULL`, porque `0` é um valor `TIMESTAMP` válido.

- Um valor padrão (`DEFAULT`) tem que ser constante, ele não pode ser uma função ou uma expressão.

  Se nenhum valor `DEFAULT` é especificado para uma coluna, o MySQL atribuirá um automaticamente, como a seguir.

  Se a coluna aceitar `NULL` como um valor, o valor padrão é `NULL`.

  Se a coluna é declarada como `NOT NULL`, o valor padrão depende do tipo de coluna:

  – Para tipos numéricos não declarados com o atributo `AUTO_INCREMENT`, o padrão é `0`. Para uma coluna `AUTO_INCREMENT`, o valor padrão é o próximo valor na sequência.

  – Para tipos date e time diferentes de `TIMESTAMP`, o padrão é o valor zero apropriado para o tipo. Para a primeira coluna `TIMESTAMP` na tabela, o padrão é a data e hora atuais. Veja Seção 6.2.2 [Tipos date e time], Página 489.

  – Para tipos string diferentes de `ENUM`, o valor padrão é uma string vazia. Para `ENUM`, o padrão é o primeiro valor enumerado.

  Valores padrões devem ser constantes. Isto significa, por exemplo, que você não pode definir o padrão de uma coluna date como o valor de funções como `NOW()` or `CURRENT_DATE`.
- Um comentário para uma coluna pode ser especificado com a opção `COMMENT`. O comentário é mostrado pela instrução `SHOW CREATE TABLE` e por `SHOW FULL COLUMNS`. Esta opção está disponível a partir do MySQL 4.1. (Ela é perimitida mas ignorada em versões anteriores.)
- `KEY` é normalmente um sinônimo para `INDEX`. A partir da versão 4.1, o atributo de chave `PRIMARY KEY` também pode ser especificado apenas como `KEY`. Isto foi implementado para compatibilidade com outros bancos de dados.
- No MySQL,uam chave `UNIQUE` só pode ter valores distintos. Um erro ocorre se você tantar adicionar uma nova linha com uma chave que coincida com uma já existente.
- `PRIMARY KEY` é uma chave única (`KEY`) onde todas as colunas chaves devem ser definidas como `NOT NULL`. Se elas não forem explicitamente declaradas como `NOT NULL`, isto será feito implicitamente e sem aviso. No MySQL a chave é chamada `PRIMARY`. Uma tabela pode ter apenas uma `PRIMARY KEY`. Se você não tiver uma `PRIMARY KEY` e alguma aplicação perguntar pela `PRIMARY KEY` em sua tabela, o MySQL retornará a primeira chave `UNIQUE`, que não possui nenhuma coluna `NULL`, como a `PRIMARY KEY`.
- Uma `PRIMARY KEY` pode ser um índice multi-coluna. Porém, você não pode criar um índice multi-coluna usando o atributo de chave `PRIMARY KEY` em uma especificação de coluna. Fazendo assim apenas colunas simples poderão ser marcadas como primárias. Você deve utilizar uma cláusula `PRIMARY KEY(index_nome_coluna, ...)` separada.
- Um índice `UNIQUE` é aquele no qual todos os valores no índice devem ser distintos. A exceção a isto é que se for permtido conter valores `NULL` em uma coluna no índice, ele pode conter múltiplos valores `NULL`. Este exceção não se aplica a tabelas `BDB`, que permitem apenas um único `NULL`.
- Se a chave `PRIMARY` ou `UNIQUE` consistir de apenas uma coluna e ela é do tipo inteiro, você também poderá se referir a ela como `_rowid` (novo na versão 3.23.11).
- Se você não atribuir um nome ao índice que não é um `PRIMARY KEY`, ele terá o mesmo nome da prmeira `index_nome_coluna`, com um sufixo opicional (`_2`, `_3`, `...`) para

torná-lo único. Você pode nome de índices para uma tabela usando `SHOW INDEX FROM nome_tabela`. Veja Seção 4.6.8.1 [Show database info], Página 303.

- Apenas os tipos de tabelas `MyISAM`, `InnoDB`, e `BDB` suportam índices em coluna que possam ter valores `NULL`. Nos outros casos você deve declarar tais colunas `NOT NULL` ou um erro será retornado.
- Com a sintaxe `nome_coluna(length)` em uma especificação de índice, você pode criar um índice que utiliza apenas os primeiros `length()` bytes de uma coluna `CHAR` ou `VARCHAR`. Isto pode tornar o arquivo de índices muito menor. Veja Seção 5.4.4 [Índices], Página 449.
- Apenas os tipos de tabela `MyISAM` e (a partir do MySQL 4.0.14) `InnoDB` suportam índice em colunas `BLOB` e `TEXT`. Ao colocar um índice em uma coluna `BLOB` ou `TEXT` você sempre DEVE especificar o tamanho do índice, até 255 bytes. Por exemplo:

```
CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));
```

- Uma especificação `index_col_name` pode finalizar com `ASC` ou `DESC`. Esta palavras chaves são permitidas para estensão futura para especificar o armazenamento do valor do índice em crescente ou decrescente. Atualmente elas são analisadas mas ignoradas; valores de índice são sempre armazenados em ordem crescente.
- Quando você utiliza `ORDER BY` ou `GROUP BY` com uma coluna `TEXT` ou `BLOB`, o servidor ardena valores usando apenas o número inicial de bytes, indicado pela variável do servidor `max_sort_length`. Veja Seção 6.2.3.2 [`BLOB`], Página 498.
- No MySQL Versão 3.23.23 ou posterior, você também pode criar índices `FULLTEXT` especiais. Eles são usados para busca full-text. Apenas o tipo de tabela `MyISAM` suporta índices `FULLTEXT`. Eles só podem ser criados em colunas `CHAR`, `VARCHAR`, e `TEXT`. A indexação sempre ocorre sobre toda a coluna; índices parciais não são suportados. Veja Seção 6.8 [Fulltext Search], Página 619 para detalhes de operação.
- No MySQL Versão 3.23.44 ou posterior, tabelas `InnoDB` suportam verificação de chaves estrangeiras. Veja Seção 7.5 [InnoDB], Página 643. Note que a sintaxe `FOREIGN KEY` no `InnoDB` é mais restrita que a sintaxe apresentada acima. As colunas da tabela indicada devem ser nomeadas explicitmente. O InnoDB suporta ambas as ações `ON DELETE` e `ON UPDATE` em chaves esrtrangiras nos MySQL 3.23.50 e 4.0.8, respectivamente. Veja a seção `InnoDB` do manual para a sintaxe precisa. Veja Seção 7.5.5.2 [InnoDB foreign key constraints], Página 654. Para outros tipos de tabelas, MySQL Server analisa as sinatxes `FOREIGN KEY`, `CHECK` e `REFERENCES` no comando `CREATE TABLE`, mas sem tal ação ser tomada. Veja Seção 1.8.4.5 [ANSI diff Foreign Keys], Página 50.
- Para tabelas `ISAM` e `MyISAM`, cada coluna `NULL` tem um bit extra, arredondado para o byte mais próximo. O tamanho máximo de um registro em bytes pode ser calculado como a seguir:

```
tamanho da linha = 1
             + (soma do tamanho da coluna)
             + (números de coluna NULL + delete_flag 7)/8
             + (número de colunas de tamanho variável)
```

`delete_flag` é 1 para tabelas com formato de registro estático. Tabelas estáticas usam um bit no registro para um parâmetro que indica se o linha foi deletada. `delete_flag` é 0 para tabelas dinâmicas porque este parâmetro é armazenado no cabeçalho da linha dinâmica.

Estes cálculos não se aplicam à tabelas `InnoDB`, para a qual o tamanho do armazenamento não é diferente para colunas `NULL` comparados a colunas `NOT NULL`.

- A opçao `opção_tabela` e `SELECT` só são implmentadas no MySQL Versão 3.23 e acima.

  A opção `TYPE` para especificar o tipo de tabela possui os seguintes valores:

Tipo de tabela	Descrição
`BDB` ou `BerkeleyDB`	Tabelas de transação segura com bloqueio de página. Veja Seção 7.6 [BDB], Página 698.
`HEAP`	Os dados desta tabela são armazenados apenas na memória. Veja Seção 7.4 [HEAP], Página 642.
`ISAM`	O mecanismo de armazenamento original. Veja Seção 7.3 [ISAM], Página 641.
`InnoDB`	Tabelas com transações eguras com bloqueio de linha. Veja Seção 7.5 [InnoDB], Página 643.
`MERGE`	Uma coleção de tabelas MyISAM usadas como uma tabela. Veja Seção 7.2 [MERGE], Página 638.
`MRG_MyISAM`	Um apelido para tabelas MERGE
`MyISAM`	O novo mecanismo de armazenamento portável binário que substitui o ISAM. Veja Seção 7.1 [`MyISAM`], Página 631.

  Veja Cap“ptexi tulo 7 [Tipos de tabelas], Página 630.

  Se um tipo de tabela é especificado, e este tipo não está disponível, MySQL irá usar `MyISAM`. Por exemplo, se uma definição de tabela inclui a opção `TYPE=BDB` mas o MySQL não suporta tabelas `BDB`, a tabela será criada como uma tabela `MyISAM`. Isto torna possível de se ter uma configuração de replicação onde você tem tabelas transacionaisno master mas as tabelas criadas no slave são não transacionais (para obter mais velocidade). No MySQL 4.1.1 você obtém um aviso se o tipo de tabela especificado não é aceito.

  Os outros tipos de tabelas são utilizados para otimizar o comportamento da tabela. Na maioria dos casos, você não precisa especificar nenhuma delas. As opções funcionam com todos os tipos, a menos que haja indicação:

Opção	Descrição
`AUTO_INCREMENT`	O próximo valor `AUTO_INCREMENT` que você quer definir em sua tabela (apenas `MyISAM`; para definir o primeiro valor auto incrementeem uma tabela InnoDB insira uma linha com um valor de menos um e delete esta linha).
`AVG_ROW_LENGTH`	Uma aproximação do tamanho médio de linha em sua tabela. Você só precisa definí-la para tabelas grnades com tamanho de registros variáveis.
`CHECKSUM`	Defina com 1 se você quiser manter um checksum para todas as linha (deixa a tabela um pouco mais lenta para atualizações, mas fica mais fácil encontrar tabelas corrompidas) (apenas `MyISAM`).
`COMMENT`	Um comentário de 60 caracteres para a sua tabela.
`MAX_ROWS`	Número máximo de linhas que você deseja armazenar na tabela.
`MIN_ROWS`	Número mínimo de linha que você planeja armazenar na tabela.

PACK_KEYS	Defina com 1 se você quiser um índice menor, Normalmente torna a atualização mais lenta e a leitura mais rápida (apenas MyISAM e ISAM). Definr com 0 irá desabilitar empacotamento das chaves. Definir com DEFAULT (MySQL 4.0) dirá ao mecanismo de armazenamento para empacotar apenas colunas CHAR/VARCHAR longas.
PASSWORD	Criptografa o arquivo '.frm' com uma senha. Esta opção não fa nada na versão padrão do MySQL.
DELAY_KEY_WRITE	Defina com 1 se quiser atrasar a atualização das chaves da tabela até que a tabela seja fechada (apenas MyISAM).
ROW_FORMAT	Define como as linhas devem ser armazenadas. Atualmente esta opção só funciona com tabelas MyISAM, as quais suportam os formatos de linha DYNAMIC e FIXED. Veja Seção 7.1.2 [Formatos das tabelas MyISAM], Página 634.

Quando você utiliza uma tabela MyISAM, MySQL usa o produto de MAX_ROWS * AVG_ROW_LENGTH para decidir o tamanho da tabela resultante. Se você não especificar qualquer uma das opções acima, o tamanho máximo de uma tabela será 4G (ou 2G se o seu sistema operacional só suporta tabelas de 2G). A razão para isto é apenas manter o tamanho dos ponteiros baixo para tornar o índice menor e mais rápido se você realmente não precisa de tabelas grandes.

Se você não utilizar PACK_KEYS, o padrão é só empacotar strings, não números. Se você utilizar PACK_KEYS=1, números também serão empacotados.

Ao empacotar chaves numéricas binárias, o MySQL usará a compactação prefixada. Isto significa que você só terá grandes benefícios disto se você tiver muitos números iguais. Compactação prefixada significa que toda a chave precisa de um byte extra para indicar quantos bytes das caves anteriores são o mesmo da próxima chave (note que o ponteiro para a linha é armazenado na ordem do byte mais alto em primeiro diretamente depois da chave, para aumentar compactação). Isto significa que se você tiver muitas chaves iguais em duas linhas consecutivas, todas os chaves "iguais" seguintes irão normalmente ter apenas 2 bytes (incluindo o ponteiro para a linha). Compare isto isto ao caso comum onde as chaves seguintes irão levar tamanho_armazenamento_chave + tamanho_ponteiro (nomralmente 4). Por outro lado, se todas as chaves são totalmente diferente, você usará 1 byte por chave, se a chave não puder ter valores NULL. (Neste caso o tamanho da chave empacotada será armazenado no mesmo byte que é usado para marcar se a chave é NULL.)

- No MySQL 3.23, Se você especificar um SELECT depois de uma instrução CREATE, MySQL criará novos campos para todos os elemento em SELECT. Por exemplo:

```
mysql> CREATE TABLE test (a INT NOT NULL AUTO_INCREMENT,
    ->        PRIMARY KEY (a), KEY(b))
    ->        TYPE=MyISAM SELECT b,c FROM test2;
```

Isto irá criar uma tabela MyISAM com três colunas, a, b e c. Note que as colunas da instrução SELECT são inseridas do lado correto da tabela, nào sobreposta nela. Considere o seguinte exemplo:

```
mysql> SELECT * FROM foo;
+---+
| n |
```

```
+---+
| 1 |
+---+

mysql> CREATE TABLE bar (m INT) SELECT n FROM foo;
Query OK, 1 row affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM bar;
+------+---+
| m    | n |
+------+---+
| NULL | 1 |
+------+---+
1 row in set (0.00 sec)
```

Para cada linha na tabela `foo`, uma linha é inserida em `bar` com os valores de `foo` e os valores padrões para a nova coluna.

`CREATE TABLE ... SELECT` não irá criar automaticamente nenhum índice para você. Isto é feito intencionalmente para deixar o comando o mais flexível possível. Se você quiser ter índices em uma tabela criada, você deve especificá-lo antes da instrução `SELECT`:

```
mysql> CREATE TABLE bar (UNIQUE (n)) SELECT n FROM foo;
```

Se ocorrer qualquer erro durante enquanto os dados são copiados para a tabela, ele será automaticamente deletado.

Você pode preceder o `SELECT` por `IGNORE` ou `REPLACE` para indicar como tratar registros que duplicam valores de chave única. Com `IGNORE`, novos registros que duplicam um registro existente em um valor de chave única são descartados. Com `REPLACE`, novos registros substituem registros que tem o mesmo valor de chave única. Se nem `IGNORE` nem `REPLACE` são especificados, valir de chave unica duplicados resultam em erro.

Para assegurar que o log binário/atualização pode ser usado para recriar a tabela original, MySQL não permitirá inserções concorrentes durante um `CREATE TABLE ... SELECT`.

- A opção `RAID_TYPE` irá ajudá-lo a exceder o limite de 2G/4G limit para arquivo de dados MyISAM (não o arquivo de índice) em sistemas operacionais que não suportam arquivos grandes. Note que esta opção não é recomendada para sistema de arquivos que suportam arquivos grandes!

  Você pode obter mais velocidade da gargalo de E/S colocando diretorios `RAID` em diferentes discos físicos. `RAID_TYPE` funcionará em qualquer sistema operacional, desde que você tenha configurado o MySQL com `--with-raid`. Por agora o único `RAID_TYPE` permitido é `STRIPED` (`1` e `RAID0` são utilizados para isto).

  Se você especificar `RAID_TYPE=STRIPED` para tabeals `MyISAM`, `MyISAM` criará subdiretórios `RAID_CHUNKS` chamados 00, 01, 02 no diretório de banco de dados. Em cada um destes diretórios `MyISAM` criará uma `nome_tabela.MYD`. Ao escrever dados no arquivo de dados, o manipulador `RAID` irá mapear o primeiro `RAID_CHUNKSIZE` *1024 bytes

para o primeiro arquivo e os próximos `RAID_CHUNKSIZE` *1024 bytes para o próximo arquivo.

- `UNION` é utilizado quando você quer utilizar uma coleção de tabelas identicas como uma. Isto só funciona com tabelas `MERGE`. Veja Seção 7.2 [`MERGE`], Página 638.

  No momento você precisa ter privilégios `SELECT`, `UPDATE` e `DELETE` nas tabelas mapeadas para uma tabela `MERGE`. Todas as tabelas mapeadas devem estar no mesmo banco de dados na tabela `MERGE`.
- Se você quiser inserir dados em uma tabela `MERGE`, você tem que especificar com `INSERT_METHOD` na tabela onde o registro deve ser inserido. `INSERT_METHOD` é uma opção útil somente para tabelas `MERGE`. Veja Seção 7.2 [MERGE], Página 638. Esta opção foi introduzida no MySQL 4.0.0.
- Na tabela criada a chave `PRIMARY` será colocado primeiro, seguida de todas a chaves únicas (`UNIQUE`) e então das chaves normais. Isto ajuda o otimizador MySQL para priorizar qual chave utilizar e também a detectaa mais rapidamente chaves únicas (`UNIQUE`) duplicadas.
- Utilizando `DATA DIRECTORY='directorio'` ou `INDEX DIRECTORY='directorio'` você pode especificar onde o mecanismo de armazenamento deve colocar os seus arquivos de tabelas e índices. Note que "diretório" deve ser um caminho completo para o diretório (não um caminho relativo).

  Isto só funciona para tabelas `MyISAM` no `MySQL` 4.0, quando não estiver usando a opção `--skip-symlink`. Veja Seção 5.6.1.2 [Links simbólicos para tabelas], Página 466.

6.5.3.1 Alteração de Especificações de Colunas

Em alguns casos, MySQL altera sem aviso uma especificação de coluna dada em uma instrução `CREATE TABLE`. (Isto também pode ocorrer com `ALTER TABLE`.):

- Colunas `VARCHAR` com um tamanho menor que quatro são alteradas para `CHAR`.
- Se qulquer coluna em uma tabela tem um tamanho variável, toda a linha é de tamanho varável como resultado. Consequentementem se uma tabela contém qualquer coluna de tamanho variável (`VARCHAR`, `TEXT`, ou `BLOB`), todas as colunas `CHAR` maior que três caracteres são alteradas para colunas `VARCHAR`. Isto não afeta como você utiliza as colunas; no MySQL, `VARCHAR` é apenas um modo diferente de armazenar caracteres. O MySQL realiza esta conversão porque ela salva espaço e torna as operções de tabela mais rápidas. Veja Cap"ptexi tulo 7 [Tipos de tabela], Página 630.
- A partir da versão 4.1.0, se um campo `CHAR` ou `VARCHAR` com uma especificação de tamanho maior que 255 é convertido para `TEXT`. Este é um recurso para compatibilidade.
- O tamanho do display `TIMESTAMP` deve ser para e na faixa de 2 a 14. Se você especificar um tamanho de display de 0 opu maior que 14, o tamaho é convertido para 14. Tamanhos de valor ímpar na faixa de 1 a 13 são convertidos para o número para mais próximo acima.
- Você não pode armazenar um `NULL` literal em uma coluna `TIMESTAMP`; definí-la com `NULL` a atribui a data e hora atual. Por colunas `TIMESTAMP` comportarem deste modo, os atributos `NULL` e `NOT NULL` não se aplicam no modo normal e são ignorados se você

especificá-los. `DESCRIBE nome_tabela` sempre indica que a uma coluna `TIMESTAMP` pode ser atribuído valores `NULL`.

- MySQL mapeia certos tipos de colunas utilizados por outros produtos de banco de dados para tipos MySQL. Veja Seção 6.2.5 [Tipos de colunas de outros produtos], Página 502.

Se você quiser ver se o MySQL utiliza um tipo de coluna diferente do especificado, axecute uma instrução `DESCRIBE nome_tabela` depois de criar ou alterar a sua tabela.

Outras alterações de tipos de colunas podem ocorrer se você compactar a tabela utilizando `myisampack`. Veja Seção 7.1.2.3 [Formato compactado], Página 635.

6.5.4 Sintaxe `ALTER TABLE`

```
ALTER [IGNORE] TABLE nome_tbl especificação_alter [, especificação_alter ...]

especificação_alter:
    ADD [COLUMN] definição_create [FIRST | AFTER nome_coluna ]
  | ADD [COLUMN] (definição_create, definição_create,...)
  | ADD INDEX [nome_indice] (index_nome_col,...)
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY (index_col_name,...)
  | ADD [CONSTRAINT [symbol]] UNIQUE [index_name] (index_col_name,...)
  | ADD FULLTEXT [index_name] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (index_col_name,...)
           [definição_referncia]
  | ALTER [COLUMN] nome_col {SET DEFAULT literal | DROP DEFAULT}
  | CHANGE [COLUMN] nome_col_antigo definição_create
              [FIRST | AFTER nome_coluna]
  | MODIFY [COLUMN] definição_create [FIRST | AFTER nome_coluna]
  | DROP [COLUMN] nome_col
  | DROP PRIMARY KEY
  | DROP INDEX nome_indice
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] nome_nova_tbl
  | ORDER BY col
  | CHARACTER SET character_set_name [COLLATE collation_name]
  |    table_options
```

`ALTER TABLE` lhe permite alterar a estrutura da tabela existente. Por exemplo, você pode adicionar ou deletar colunas, criar ou remover índices, alterar o tipo de coluna existentes, ou renomear coluna ou tabelas. Você também pode alterar o comentário para a tabela e tipo de tabela. Veja Seção 6.5.3 [`CREATE TABLE`], Página 598.

Se você utilizar `ALTER TABLE` para alterar a especificação da coluna, mas `DESCRIBE tbl_name` indicar que a sua coluna não foi alterada, é possível que o MySQL tenha ignorado ou a sua modificação por uma das razões descritas em Seção 6.5.3.1 [Silent column changes], Página 607. Por exemplo, se você tentar alterar uma coluna `VARCHAR` para `CHAR`, MySQL ainda usará `VARCHAR` se a tabela conter outras colunas de tamanho variável.

`ALTER TABLE` funciona fazendo uma cópia temporária da tabela original. A alteração é realizada na cópia, assim a tabela original é deletada e a nova tabela é renomeada. Isto é feito de tal forma que todas as desnecessáriaatualizações são automaticamente redirecionadas para a nova tabela sem nenhuma atualização errada. Enquanto o `ALTER TABLE` é executado, a tabela original pode ser lida por outros clientes. Atualizações e escrita na tabela são guardadas até a nova tabela estar pronta.

Note que se você utilizar qualquer outra opção de `ALTER TABLE`, exceto `RENAME`, o MySQL irá sempre criar um a tabela temporária, mesmo se os dados não precisarem realmente serem copiados (como quando você altera o nome de uma coluna). Planejamos corrigir isto no futuro, mas como não se faz `ALTER TABLE` com tanta frequência, isto não é de alta prioridade em nosso TO DO. Para tabelas MyISAM, vOcê pode aumentar a velocidade na parte da recriação dos índices (que a parte mais lenta do processo recriação) atribuindo um alto valor à variável `myisam_sort_buffer_size`.

- Para utilizar `ALTER TABLE`, você precisa dos privilégios `ALTER`, `INSERT` e `CREATE` na tabela.
- `IGNORE` é uma extensão do MySQL ao SQL-92. Ele controla como o `ALTER TABLE` funciona se houver duplicação em chaves únicas na nova tabela. Se `IGNORE` não é especificado, a cópia é abortada e retornada. Se `IGNORE` for especificado, para linhas com duplicatas em chaves únicas, somente a primera linha é usada; as outras são deletadas.
- Você pode executar múltiplas cláusulas `ADD`, `ALTER`, `DROP` e `CHANGE` em uma única instrução `ALTER TABLE`. Esta é uma extensão do MySQL ao SQL-92, que permite paenas uma cláusula de cada por instrução `ALTER TABLE`.
- `CHANGE col_name`, `DROP col_name`, e `DROP INDEX` são extensões do MySQL ao SQL-92.
- `MODIFY` é uma extensão do Oracle para `ALTER TABLE`.
- A palavra opcional `COLUMN` é uma palavra puramente desnecessária e pode ser omitida.
- Se você utilizar `ALTER TABLE nome_tbl RENAME TO novo_nome` sem nenhuma outra opção, MySQL simplesmente renomeia os arquivos correspondentes a tabela `nome_tbl`. Não há necessidade de se criar uma tabela temporária. Veja Seção 6.5.5 [`RENAME TABLE`], Página 612.
- Cláusulas `definição_create` usam a mesma sintaxe para `ADD` e `CHANGE` assim como para `CREATE TABLE`. Note que a sintaxe inclui o nome da coluna, não apenas o tipo da coluna. Veja Seção 6.5.3 [`CREATE TABLE`], Página 598.
- Você pode renomear ma coluna usando uma cláusula `CHANGE nome_col_antiga definições_create`. Para tal, especifique o nome das colunas antiga e da nome e o tipo que a coluna atual possui. Por exemplo, para renomear uma coluna `INTEGER` de `a` para `b`, faça assim:

  ```
  mysql> ALTER TABLE t1 CHANGE a b INTEGER;
  ```

  Se você quiser mudar um tipo de coluna, mas não o nome, a sintaxe `CHANGE` ainda exige dois nomes de colunas, mesmo que sejam o mesmo. Por exemplo:

  ```
  mysql> ALTER TABLE t1 CHANGE b b BIGINT NOT NULL;
  ```

  No entanto, como no MySQL Versão 3.22.16a, você também pode utilizar `MODIFY` para alterar um tipo de coluna sem renomeá-lo:

  ```
  mysql> ALTER TABLE t1 MODIFY b BIGINT NOT NULL;
  ```

- Se você utilizar `CHANGE` ou `MODIFY` para reduzir uma coluna na qual exista um índice em parte da coluna (por exemplo, se você tiver um índice nos primeiros 10 caracteres de uma coluna `VARCHAR`), você não poderá reduzir a coluna para um tamanho menor que o número de caracteres indexados.
- Quando você altera um tipo de coluna usando `CHANGE` ou `MODIFY`, erter os dados para o novo tipo da melhor forma possível.
- No MySQL Versão 3.22 ou posterior você pode utilizar `FIRST` ou `ADD ... AFTER nome_col` para aadicionar uma coluna em uma posição específica na linha da tabela. O padrão é adicionar a coluna no fim. A partir do MySQL Versão 4.0.1, você pode também utilizar as palavras-chave `FIRST` e `AFTER` em `CHANGE` ou `MODIFY`.
- `ALTER COLUMN` especifica um novo valor padrão para uma coluna ou remover o valor padrão antigo. Se o padrão antigo é removido e a coluna pode ser `NULL`, o novo padrão é `NULL`. Se a coluna não pode ser `NULL`, MySQL atribui um valor padrão, como descrito em Seção 6.5.3 [`CREATE TABLE`], Página 598.
- `DROP INDEX` remove um índice. Esta é uma extensão do MySQL ao SQL-92. Veja Seção 6.5.8 [DROP INDEX], Página 614.
- Se colunas forem removidas de uma tabela, as colunas também são removidas de qualquer índice do qual eles fazem parte. Se todas as colunas que compõe um índice são excluídas, o índice também é excluído.
- Se uma tabela contém apenas uma coluna, a coluna não pode ser excluída. Se o que você pretende é remover a tabela, use `DROP TABLE`.
- `DROP PRIMARY KEY` deleta o índice primário. Se tal índice não existe, ele apaga o prmeiro índice único (`UNIQUE`) na tabela. (MySQL marca a primeira chave única (`UNIQUE`) como `PRIMARY KEY` se nenhuma `PRIMARY KEY` foi especificada explicitamente.)

  Se você adicionar `UNIQUE INDEX` ou `PRIMARY KEY` a uma tabela, elas são armazenadas antes de qualquer índice não `UNIQUE` para que possa detectar cahves duplicadas o mais rápido possível.
- `ORDER BY` lhe permite criar a nova tabela com as linhas em uma ordem específica. Note que a tabela não permanecerá nesta ordem depois de insrções e deleções. Em algunas casos, isto pode tornar a ordenação mais para o MySQL se a tabela estiver ordenada pela coluna que você escolheu. Esta opção é útil principalmente quando você sabe qeu na maioria das vezes você irá inserir os registros em certa ordem; utilizando esta opção depois de grandes mudanças na tabela, você obterá melhor desempenho.
- Se você utilizar `ALTER TABLE` em uma tabela `MyISAM`, todos os índices que não são únicos são criados em um grupo separado (como em `REPAIR`). Isto deve tornar `ALTER TABLE` muito mais rápido quando você tiver vários índices.
- A partir do **MySQL 4.0** o recurso acima pode ser ativado explicitamente. `ALTER TABLE ... DISABLE KEYS` faz o MySQL parar de atualizar chaves que não são únicas em tabelas `MyISAM`. `ALTER TABLE ... ENABLE KEYS` deve ser usado para recriar índices perdidos. Como o MySQL faz isso com um algoritmo especial que é muito mais rápido que inserir chaves uma a uma, disabilitar chaves podem trazer um aumento de velocidade considerável em inserções volumosas.
- Com a função `mysql_info()` da API C, você pode saber quantos registros foram copiados, e (quando `IGNORE` for usado) quantos registros foram deletados devido a duplicação de valores de chaves únicas.

- As cláusulas `FOREIGN KEY`, `CHECK` e `REFERENCES` não fazem nada, exceto para tipos de tabela InnoDB que suportam `... ADD [CONSTRAINT [symbol]] FOREIGN KEY (...) REFERENCES ... (...)` e `... DROP FOREIGN KEY ...`. Veja Seção 7.5.5.2 [InnoDB foreign key constraints], Página 654. A sintaxe para outros tipos de tabela só é fornecido para comptibilidade, para tornar fácil portar o código de outro servidor SQL e executar aplicações que criam tabelasd com referências. Veja Seção 1.8.4 [Diferenças do ANSI], Página 46.
- `ALTER TABLE` ignora as opções de tabela `DATA DIRECTORY` e `INDEX DIRECTORY`.
- Se você quiser alterar todas as colunas CHAR/VARCHAR/TEXT para um novo conjunto de caracteres (por exemplo, depois de atualizar do MySQL 4.0.x para o 4.1.1) você pode fazer:

  ```
  ALTER TABLE table_name CHARACTER SET character_set_name;
  ```

  Note que o seguinte comando só irá alterar o `default character set` para uma tabela:

  ```
  ALTER TABLE table_name DEFAULT CHARACTER SET character_set_name;
  ```

  O `default character set` é o conjunto de caracteres que é usado se você não especificar o conjunto de caracteres para uma nova coluna que você adicionar a tabela (por exemplo com `ALTER TABLE ... ADD coluna`).

Aqui temos um exemplo que mostra alguns dos usos de `ALTER TABLE`. Nós começamos com uma tabela `t1` que é crida como mostrado aqui:

```
mysql> CREATE TABLE t1 (a INTEGER,b CHAR(10));
```

Para renomear a tabela de `t1` para `t2`:

```
mysql> ALTER TABLE t1 RENAME t2;
```

Para alterar a coluna `a` de `INTEGER` para `TINYINT NOT NULL` (deixando o mesmo nome), e alterar a coluna `b` de `CHAR(10)` para `CHAR(20)` e renomeá-la de `b` para `c`:

```
mysql> ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);
```

Para adicionar um nova coluna `TIMESTAMP` chamada `d`:

```
mysql> ALTER TABLE t2 ADD d TIMESTAMP;
```

Para adicionar um índice na coluna `d`, e tornar a colua `a` a chave primária:

```
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);
```

Para remover a coluna `c`:

```
mysql> ALTER TABLE t2 DROP COLUMN c;
```

Para adiciomar um nova coluna inteira `AUTO_INCREMENT` chamada `c`:

```
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
           ADD INDEX (c);
```

Note que nós indexamos `c`, porque colunas `AUTO_INCREMENT` devem ser indexadas e também por isso declaramos `c` como `NOT NULL`, pois colunas indexadas não podem ser `NULL`.

Quando você adicionar uma coluna `AUTO_INCREMENT`, valores de coluna são preenchidos com sequência de números automaticamente para você. Você pode definir o primeiro número da sequência executando `SET INSERT_ID=valor` antes de `ALTER TABLE` ou usando a opção de tabela `AUTO_INCREMENT=valor`. Veja Seção 5.5.6 [SET OPTION], Página 460.

Com tabelas MyISAM tables, se você não alterar a coluna `AUTO_INCREMENT`, a sequência de números não será afetada. Se você excluir uma coluna `AUTO_INCREMENT` e adicionar outra coluna `AUTO_INCREMENT`, a numeração iniciará a partir do 1 novamente.

Veja Seção A.7.1 [Problemas com ALTER TABLE], Página 938.

6.5.5 Sintaxe `RENAME TABLE`

```
RENAME TABLE nome_tabela TO novo_nome_tabela[, nome_tabela2 TO novo_nome_tbl2,...]
```

A renomeação é feita automicamente, o que significa que nenhuma outra thread pode acessar qualquer uma das tabelas enquanto a renomeação está sendo exectuda. Isto torna possível substituir uma tabela por uma tabela vazia:

```
CREATE TABLE tabela_nova (...);
RENAME TABLE tabela_antiga TO tabela_backup, tabela_nova TO tabela_antiga;
```

A renomeação é feita da esquera para a direita, o que significa que se você quiser trocar os nomes das tabelas, você deve fazer:

```
RENAME TABLE tabela_antiga TO tabela_backup,
             tabela_nova   TO tabela_antiga,
             tabela_backup TO tabela_nova;
```

Desde que dois banco de dados estejam no mesmo disco você pode renomear de um banco de dados para outro:

```
RENAME TABLE bd_atual.nome_tabela TO outro_bd.nome_tabela;
```

Quando você executa `RENAME`, você não pode ter nenhuma tabela bloqueada ou transações ativas. Você também deve ter o privilégio `ALTER` e `DROP` na tabela original e o privilégio `CREATE` e `INSERT` na nova tabela.

Se o MySQL encontrar qualquer erro uma renomeação multi-tabela, ele fará um renomeação reversa para todas a tabelas renomeadas para retornar tudo ao estado original.

`RENAME TABLE` foi adicionado no MySQL 3.23.23.

6.5.6 Sintaxe `DROP TABLE`

```
DROP [TEMPORARY] TABLE [IF EXISTS] nome_tabela [, nome_tabela,...] [RESTRICT | CASC
```

`DROP TABLE` remove uma ou mais tabelas. Todos os dados e definições de tabela são *removidos*, assim **tenha cuidado** com este comando!

No MySQL Versão 3.22 ou posteriorm você pode usar a palavra-chave `IF EXISTS` para prevenir um erro de ocorrer se não existir a tabela. Na versão 4.1 consegue-se um `NOTA` para todas as tabelas não esistentes se for usado `IF EXISTS`. Veja Seção 4.6.8.9 [SHOW WARNINGS], Página 323.

`RESTRICT` e `CASCADE` são permitidos para portação se tornar tornar mais fácil. No momento eles não fazem nada.

Nota: `DROP TABLE` fará automaticamente um commit da transação ativa atualmente (exceto se você estiver usando a versão 4.1 e a palavra-chave `TEMPORARY`.

A opcão `TEMPORARY` é ignorada na versão 4.0. Na versão 4.1 esta opção funciona como a seguir:

- Só apaga tabelas temporárias.
- INão finaliza uma transação em execução.
- Nenhum direito de acesso é verificado.

Usar `TEMPORARY` é uma boa maneira de assegurar que você não apague uma tabela real.

6.5.7 Sintaxe `CREATE INDEX`

```
CREATE [UNIQUE|FULLTEXT] INDEX nome_indice
       ON nome_tabela (index_col_name,...)

index_col_name:
       col_name [(length)] [ASC | DESC]
```

A instrução `CREATE INDEX` não faz nada em versões do MySQL anterior a 3.22. Na versão 3.22 ou posteriores, `CREATE INDEX` é mapeado para uma instrução `ALTER TABLE` para criar índices. Veja Seção 6.5.4 [`ALTER TABLE`], Página 608.

Normalmente você cria todos os índices em uma tabela ao mesmo tempo em que a própria tabela é criada com `CREATE TABLE`. Veja Seção 6.5.3 [`CREATE TABLE`], Página 598. `CREATE INDEX` lhe permite adicionar índices a tabelas existentes.

Uma lista de colunas na forma `(col1,col2,...)` cria um índice com múltiplas colunas. Valores de índice são formados concatenando os valores de colunas dadas.

Para colunas `CHAR` e `VARCHAR`, índices que utilizam apenas parte da coluna podem ser criados, usando a sintaxe `nome_coluna(length)` para indexar os primeiros `length()` bytes de cada valor da coluna. (Para colunas `BLOB` e `TEXT`, um prefixo length é exigido; `length()` pode ter um valor até 255 caracteres.) A instrução mostrada aqui cria um índice usando os primeiros 10 caracteres da coluna `name`:

```
mysql> CREATE INDEX part_of_name ON customer (name(10));
```

Como a maioria dos nomes normalmente diferem nos primeiros 10 caracteres, este índice não deve ser muito menor que um índice criado com toda a coluna `name`. Além disso, usar colunas parciais como índices pode fazer o arquivo de índice muito menor, o que pode economizar muito espaço em disco e pode também aumentar a velocidade de operações `INSERT`!

Note que você pode adicionar um índice em uma coluna que pode ter valores apenas se você estiver usando o MySQL Versão 3.23.2 ou mais novo e estiver usando os tipos de tabelas `MyISAM`, `InnoDB`, ou `BDB`. Você só pode adicionar um índice em uma coluna `BLOB/ TEXT` se você estiver usando o MySQL Versão 3.23.2 ou mais novo e estiver usando os tipos de tablea `MyISAM` ou `BDB`, ou MySQL Versão 4.0.14 ou mais novo e o tipo de tabela `InnoDB`. Para um índice em uma coluna `BLOB/TEXT`, o tamanho do prefixo sempre deve ser especificado.

Uma especificação `index_col_name` pode finalizar com `ASC` ou `DESC`. Esta palavras chaves são permitidas para estensão futura para especificar o armazenamento do valor do índice em crescente ou decrescente. Atualmente elas são analisadas mas ignoradas; valores de índice são sempre armazenados em ordem crescente.

Para mais detalhes sobre como o MySQL utiliza índices, veja Seção 5.4.3 [Índices MySQL], Página 447.

Indíces FULLTEXT só podem indexar colunas CHAR, VARCHAR e TEXT, e apenas em tabelas MyISAM. Índices FULLTEXT estão disponíveis no MySQL Versão 3.23.23 e posterior. Seção 6.8 [Pesquisa Fulltext], Página 619.

6.5.8 Sintaxe DROP INDEX

```
DROP INDEX nome_indice ON nome_tabela
```

DROP INDEX apaga o índice chamado nome_indice da tabela nome_tabela. DROP INDEX não faz nada nem versões do MySQL anteriores a 3.22. Na versão 3.22 ou posterior, DROP INDEX é mapeada em uma instrução ALTER TABLE para apagar o índice. Veja Seção 6.5.4 [ALTER TABLE], Página 608.

6.6 Comandos Utilitários Básicos do Usuário MySQL

6.6.1 Sintaxe USE

```
USE nome_db
```

A instrução USE nome_bd diz ao MySQL para usar o banco de dados nome_bd como padrão para as consultas subsequentes. O banco de dados continua como o atual até o final da sessão ou até outra instrução USE ser executada:

```
mysql> USE db1;
mysql> SELECT COUNT(*) FROM mytable;      # seleciona de  db1.mytable
mysql> USE db2;
mysql> SELECT COUNT(*) FROM mytable;      # seleciona de db2.mytable
```

Torna um banco de dados particular como o atual não significa que a instrução USE não o permita acessar tabelas em outros bancos de dados. O exemplo seguinte acessa a tabela author do banco de dados db1 e a tabela editor do banco de dados db2:

```
mysql> USE db1;
mysql> SELECT author_name,editor_name FROM author,db2.editor
    ->        WHERE author.editor_id = db2.editor.editor_id;
```

A instrução USE é fornecida para compatibilidade com o Sybase.

6.6.2 Sintaxe DESCRIBE (Obtem Informações Sobre Colunas)

```
{DESCRIBE | DESC} nome_tabela [nome_coluna | meta_carac]
```

DESCRIBE á um atalho para SHOW COLUMNS FROM. Veja Seção 4.6.8.1 [Show database info], Página 303.

DESCRIBE fornece informação sobre as colunas da tabela. nome_coluna deve ser um nome de coluna ou uma string contendo os meta caracteres '%' e '_' do SQL para ter a saída apenas com nomes que corespondam com a string. Não é necessário colocar a string entre aspas.

Se os tipos de colunas são diferentes do esperado baseado nas instruções CREATE TABLE, note que algumas vezer o MySQL altera o tipo das colunas. Veja Seção 6.5.3.1 [Alterando colunas sem aviso], Página 607.

Esta instrução é fornecida para compatibilidade com Oracle.

A instrução `SHOW` fornece informação similar. Veja Seção 4.6.8 [SHOW], Página 302.

6.7 Comandos Transacionais e de Lock do MySQL

6.7.1 Sintaxe de `START TRANSACTION`, `COMMIT` e `ROLLBACK`

Por padrão, MySQL é executado em modo autocommit. Isto significa que assim que você executa uma instrução que atualiza (modifica) uma tabela, o MySQL armaena a atualização no disco.

Se você estiver usando tabelas com segurança a transação (como `InnoDB \` ou `BDB`), você pode colocar o MySQL em modo não autocommit com o seguinte comando:

```
SET AUTOCOMMIT=0
```

Depois de disabilitar o modo autocommit configurando a variável `AUTOCOMMIT` com zero, você deve utilizar `COMMIT` para armazenar suas alterações em disco ou `ROLLBACK` se você deseja ignorar as alterações que você fez desde o início da sua transação.

Se você quiser disabilitar o modo autocommit para uma única série de instruções, você pode utiliar a instrução `START TRANSACTION`:

```
START TRANSACTION;
SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summmary=@A WHERE type=1;
COMMIT;
```

`BEGIN` e `BEGIN WORK` podem ser usados em vez de `START TRANSACTION` para iniciar uma transação. `START TRANSACTION` foi adicionado no MySQL 4.0.11; ele é uma sintaxe do SQL-99 e é o modo recomendado de iniciar umaa transação an ad-hoc. `BEGIN` e `BEGIN WORK` estão disponíveis a partir do MySQL 3.23.17 e 3.23.19, respectivamente.

Note que se você estiver usando tabelas sem segurança a transação, quaisquer alterações serão armazenadas de uma vez, se considerar o status do modo autocommit.

Se você executar uma instrução `ROLLBACK` depois de atualizar uma tabela não-transacional, você obterá um erro (`ER_WARNING_NOT_COMPLETE_ROLLBACK`), como um aviso. Todas as tabelas seguras a transação serão restauradas mas qualquer tabela se segurança a transação não sofrerão alterações.

Se você estiver usando `START TRANSACTION` ou `SET AUTOCOMMIT=0`, você deve usar o log binário do MySQL para backup no lugar do antigo log de atualização. Transações são armazenadas no log binário em um bloco, sobre `COMMIT`, para assegurar que transações nas quais foram feitas rolled back não foram armazenadas. Veja Seção 4.10.4 [Log binário], Página 374.

Você pode alterar o nível isolação para transações com `SET TRANSACTION ISOLATION LEVEL`. Veja Seção 6.7.6 [SET TRANSACTION], Página 619.

6.7.2 Instruções que Não Podem Ser Desfeitas

Não se pode fazer o roll back de algumas instruções. Em geral, elas incluem instruções DDL (data definition language), como aquelas que criam ou removem banco de dados, ou aquelas que criam, apagam ou alteram tabelas.

Você pode desejar projetar as suas transações para não incluir estas instruções. Se você executar uma instrução da quale não se pode fazer roll back em uma transação, e então outra intruções falhar posteriormente, o efeito total da transação não pode ser desfeito usando uma instrução `ROLLBACK`.

6.7.3 Instruções que Fazem um Commit Implicito

Os seguintes comandos finalizam uma transação implicitamente (como se você tivesse feito um `COMMIT` antes de executar o comando):

Comando	**Comando**	**Comando**
`ALTER TABLE`	`BEGIN`	`CREATE INDEX`
`DROP DATABASE`	`DROP INDEX`	`DROP TABLE`
`LOAD MASTER DATA`	`LOCK TABLES`	`RENAME TABLE`
`SET AUTOCOMMIT=1`	`START TRANSACTION`	`TRUNCATE`

`UNLOCK TABLES` também finaliza uma transação se qualquer tabela estiver atualmente bloqueada. Antes do MySQL 4.0.13, `CREATE TABLE` finaliza uma transação se o log binário está habilitado.

Transações não podem ser aninhadas. Isto é uma consequência do `COMMIT` implícito realizado por qualquer transação atual quando você envia uma instrução `START TRANSACTION` ou um de seus sinônimos.

6.7.4 Sintaxe de `SAVEPOINT` e `ROLLBACK TO SAVEPOINT`

A partir do MySQL 4.0.14 e 4.1.1. o `InnoDB` suporta os comando SQL `SAVEPOINT` e `ROLLBACK TO SAVEPOINT`.

```
SAVEPOINT identificador
```

Esta instrução configura um savepoint de uma transação cujo nome é `identificador`. Se a transação atual já tiver um savepoint com o mesmo nome, o savepointy antigo é deletado é o novo é definido.

```
ROLLBACK TO SAVEPOINT identificador
```

Esta instrução faz o roll back de uma transação até o savepoint indicado. Modificações feitas nesta transação após o savepoint foram definidas como desfeitas no roll back, mas o `InnoDB` **não** libera o lock de linha que forma arnmazenados na memória depois do savepoint. (Note que para uma nova linha inserida, a informação do lock é carregada pala ID da transação armazenada na linha; o lock não é armazenado separadamente na memória. Neste caso, o lock de linha é liberado no undo.) Sevapoints que foram definidos após o sevepoint indicado são deletados.

Se o comando retorna o seguinte erro, significa que não existem savepoints como o nome especificado.

```
ERROR 1181: Got error 153 during ROLLBACK
```

Todos os savepoints da transação atual são deletados se você executar um `COMMIT` ou um `ROLLBACK` que não chamou um savepoint.

6.7.5 Sintaxe `LOCK TABLES` e `UNLOCK TABLES`

```
LOCK TABLES nome_tabela [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE}
            [, nome_tabela [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE} ...]
...
UNLOCK TABLES
```

`LOCK TABLES` bloqueia tabelas para a thread atual. `UNLOCK TABLES` libera qualquer trava existente para a thread atual. Todas as tabela que estão bloqueadas pela thread atual são implicitamente desbloquadas quando a thread executa um outro `LOCK TABLES`, ou quando a conexão ao servidor é fechada.

Para usar `LOCK TABLES` no MySQL 4.0.2 você precisa do privilégio global `LOCK TABLES` e um privilégio `SELECT` nas tabelas envolvidas No MySQL 3.23 você precisa ter os privilégios `SELECT`, `insert`, `DELETE` e `UPDATE` para as tabelas.

A razão principal para utilizar `LOCK TABLES` é para emular transações ou obter mais velocidade ao atualizar tabelas. Isto é explicado em mais detalhes posteriormente.

Se uma thread obtem uma trava de leitura (`READ`) em uma tabela, aquela thread (e todas as outras threads) só poderão ler da tabela. Se uma thread obter uma trava de escrita (`WRITE`) na tabela, apenas a thread que bloqueou poderá ler ou escrever na tabela. Outras threads serão bloqueadas.

A diferença entre `READ LOCAL` e `READ` é que `READ LOCAL` permite que instruções `INSERT` não conflitantes sejam executadas enquanto a trava está ativa. Isto, no entatnto, não pode ser usado se você for manipular o arquivo de banco de dados fora do MySQL enquanto a trava estiver ativa.

Quando você usa `LOCK TABLES`, você deve travar todas as tabelas que você for usar e utilizar o mesmo alias que estiver utilizando em suas consultas! Se você estiver usando uma tabela várias vezes em uma consulta (com aliases), você deve obter um trava para cada alias.

Bloqueio de escrita (`WRITE`) normalmente têm maior prioridade que bloqueio de leitura (`READ`), para assegurar que atualizações são processadas assim que possível. Isto significa que se uma thread obtida um bloqueio de leitura (`READ`) e outra thread requisitar um bloqueio de escrita (`WRITE`), bloqueios de leitura (`READ`) subsequentes irão esperar até a thread de escrita (`WRITE`) tiver obtido a trava e a liberado. Você pode usar travas `LOW_PRIORITY WRITE` para permitir que outras threads obtenham bloqueios de leitura (`READ`) enquanto a thread estiver esperando pela trava de escrita (`WRITE`). Você só deve utilizar bloqueios `LOW_PRIORITY WRITE` se você estiver certo que haverá um momento onde nenhuma thread terá bloqueio de leitura (`READ`).

`LOCK TABLES` funciona da seguinte maneira:

1. Ordene todas as tabelas a serem travadas em uma ordem definida internamente (do ponto do usuário a ordem é indefinida).
2. Se uma tabela é bloqueada com uma trava de leitura e de escrita, coloque a trava de escrita antes da trava de leitura.

3. Bloqueie uma tabela por vez até que a thread obtenha todas as travas.

Esta polítca assegura que as tabelas sejam bloqueadas sem deadlock. Há no entanto outra coisa da qual é preciso estar ciente neste esquema:

Se cocê estiver usando uma trava de escita `LOW_PRIORITY WRITE` em uma tabela, significa apenas que o MySQL irá esperar por esta trava particular até que não haja mais treads fazendo um bloqueio de leitura (`READ`). Quando a thread tiver obtido a trava de escrita (`WRITE`) e está esperando ppo obter o trava para a próxima tabela na lista de tabelas bloqueadas, todas as outras threads irão esperar que a trva de escrita (`WRITE`) seja liberada. Se isto tornar um sério problema com sua aplicação, você deve converter algumas de suas tabellas para tabelas com segurança em transações.

Você pode matar com segurança um thread que está esperando por um bloqueio de tabela com `KILL`. Veja Seção 4.6.7 [KILL], Página 301.

Note que você **não** deve travar nenhuma tabela que você esteja usando com `INSERT DELAYED`. Isto é porque este é o caso que o `INSERT` é feito por uma thread separada.

Normalmente, você não tem que travar tabelas, já que todas as instruções `UPDATE` são atomicas; nenhuma outra thread pode interferir com qualquer outra executando uma instrução SQL. Existem poucos casos em que você gostaria de travar as tabelas de qualquer forma:

- Se você for executar operações em um grupo de tabelas, é muito mais rápido travar as tabelas que você for utilizar. O lado ruim é que nenhuma outra thread pode atualizar uma tabela travada para leitura (`READ`) (incluindo aquela que guarda o lock) e nenhuma outra thread pode ler uma tabela bloqueada para escrita (`WRITE`) além daquele que guarda o lock.

  A razão de algumas coisas serem rápidas sob `LOCK TABLES` é que o MySQL não irá descarregar a cache de tabelas bloqueadas até que `UNLOCK TABLES` seja chamado (normalmente a cache de chaves é descarregada a cada instrução SQL). Isto aumenta a velocidade de inserção, atualização e deleção) em tabelas `MyISAM`.

- Se você estiver usando um mecanismo de armazenamento no MySQL que não suporte transações, você deve usar `LOCK TABLES` se você quiser se assegurar que nenhuma outra thread venha entre um `SELECT` e um `UPDATE`. O exemplo mostrado aqui exige `LOCK TABLES` para ser executado com segurança:

  ```
  mysql> LOCK TABLES trans READ, customer WRITE;
  mysql> SELECT SUM(value) FROM trans WHERE customer_id=some_id;
  mysql> UPDATE customer SET total_value=sum_from_previous_statement
      ->        WHERE customer_id=some_id;
  mysql> UNLOCK TABLES;
  ```

  Sem `LOCK TABLES`, existe uma chance que outra thread possa inserir uma nova linha na tabela `trans` entre a execução das instruções `SELECT` e `UPDATE`.

Utilizando atualizações incrementais (`UPDATE customer SET value=value+new_value`) ou a função `LAST_INSERT_ID()`i, você pode evitar o uso de `LOCK TABLES` em muitos casos.

Você também pode resolver alguns casos usando as funções de bloqueio a nível de usuário `GET_LOCK()` e `RELEASE_LOCK()`. Estas travas são salvas em uma tabela hash no servidor e implementado com `pthread_mutex_lock()` e `pthread_mutex_unlock()` para alta velocidade. Veja Seção 6.3.6.2 [Funções diversas], Página 547.

Veja Seção 5.3.1 [Internal locking], Página 443, para mais informações sobre política de bloqueios.

Você pode trocar todas as tabelas em todos os banco de dados com trava de leitura com o comando `FLUSH TABLES WITH READ LOCK`. Veja Seção 4.6.4 [FLUSH], Página 299. Este é um modo muito conveiente de tirar backups se você tiver um sistema de arquivos, como Veritas, que pode tirar snapshots.

NOTE: `LOCK TABLES` mão é seguro com transações e fará um commit implicitamente em qualquer transação ativa antes de tentar travar as tabelas.

6.7.6 Sintaxe `SET TRANSACTION`

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
{ READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }
```

Define o nível de isolação da transação para global, toda a sessão ou a próxima transação.

O comportamento padrão é definir o nível de isolação para a próxima (não iniciada) transação. Se você usa a palavra-chave `GLOBAL`, a instrução define o nivel de transação padrão globalmente para todas as novas conexões criadas a partir deste ponto (mas não existe conexão). Você precisa do privilégio `SUPER` para fazer isto. Usar a palavra-chave `SESSION` define o nível de transação padrão para todas a transações futuras relaizadas na conexão atual.

Para a descrição de cada nível de isolação da transação do `InnoDB`, veja Seção 7.5.9.1 [InnoDB transaction isolation], Página 662. O InnoDB suporta cada um destes níveis a partir do MySQL 4.0.5. O nível padrão é `REPEATABLE READ`.

Você pode definir o nível de isolação global padrão para o `mysqld` com `--transaction-isolation=...`. Veja Seção 4.1.1 [Opções de linha de comando], Página 207.

6.8 Pesquisa Full-text no MySQL

```
MATCH (col1,col2,...) AGAINST (expr [IN BOOLEAN MODE | WITH QUERY EXPANSION] )
```

A partir da versão 3.23.23, MySQL tem suporte para indexação e busca full-text. Índices full-text no MySQL são um índice do tipo `FULLTEXT`. Índices `FULLTEXT` são usados apenas com tabelas `MyISAM` e podem ser criadas a partir de colunas `CHAR`, `VARCHAR` ou `TEXT` durante um `CREATE TABLE` ou adicionados posteriormente com `ALTER TABLE` ou `CREATE INDEX`. Para banco de dados maiores, será muito mais rápido carregar seus dados em uma tabela que não tnha índices `FULLTEXT`, que criar o índice com `ALTER TABLE` (ou `CREATE INDEX`). Carregar dados em uma tabela que já tenha um índice `FULLTEXT` será muito mais lento.

Pesquisa full-text é realizada com a função `MATCH()`.

```
mysql> CREATE TABLE articles (
    ->   id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    ->   title VARCHAR(200),
    ->   body TEXT,
    ->   FULLTEXT (title,body)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> INSERT INTO articles VALUES
    -> (NULL,'MySQL Tutorial', 'DBMS stands for DataBase ...'),
    -> (NULL,'How To Use MySQL Efficiently', 'After you went through a ...'),
    -> (NULL,'Optimizing MySQL','In this tutorial we will show ...'),
    -> (NULL,'1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    -> (NULL,'MySQL vs. YourSQL', 'In the following database comparison ...'),
    -> (NULL,'MySQL Security', 'When configured properly, MySQL ...');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM articles
    ->          WHERE MATCH (title,body) AGAINST ('database');
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
+----+-------------------+------------------------------------------+
2 rows in set (0.00 sec)
```

A função `MATCH()` realiza um busca de linguagem natural por uma string contra uma coleção de texto (um conjunto de uma ou mais colunas incluídas em um índice `FULLTEXT`). A string pesquisada é dada como o argumento de `AGAINST()`. A busca é realizada na forma caso-insensitivo. Para cada uma das linhas da tabela, `MATCH()` retorna um valor relevante, isto é, uma medida de similaridade entre a string pesquisada e o texto naquela nas colunas identificadas na lista `MATCH()`.

Quando `MATCH()` é utilizado na cláusula `WHERE` (veja exemplo acima) as linhas retornadas são automaticamente ordenadas com a maior relevância primerio. Valores de relevância são números de ponto flutuante não negativos. Relevância zero significa nenhuma similaridade. Relevância é computado baseada no número de palavras na linha, o número de palavras única naquela linha, o número de palavras na coleção e o número de documentos (linhas) que contenham uma palavra particular.

Também é possível realizar uma busca no modo booleano. Isto é explicado posteriormente nesta seção.

O exemplo precedente é uma ilustrção básica mostrando como usar a função `MATCH()`. Linhas são retornodas em ordem decrescente de relevância.

O próximo exemplo mostra como retornar o valores de relevância explicitamente. Como nem a cláusula `WHERE` nem a `ORDER BY` estão presentes, as linhas são retornadas fora de ordem.

```
mysql> SELECT id,MATCH (title,body) AGAINST ('Tutorial') FROM articles;
+----+----------------------------------------+
| id | MATCH (title,body) AGAINST ('Tutorial') |
+----+----------------------------------------+
|  1 |                   0.64840710366884 |
|  2 |                                  0 |
```

```
|  3 |                         0.66266459031789 |
|  4 |                                        0 |
|  5 |                                        0 |
|  6 |                                        0 |
+----+-----------------------------------------+
6 rows in set (0.00 sec)
```

O exemplo seguinte é mais complexo. A consulta retorna a relevância e ainda ordena as linhas em ordem decrescente de relevância. Para conseguir este resultado, você deve especificar `MATCH()` duas vezes. Isto não irá causar sobrecarga adicional, pois o otimizador MySQL irá notar que duas chamadas `MATCH()` são idênticas e invocam o código da busca full-text apenas uma vez.

```
mysql> SELECT id, body, MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root') AS score
    -> FROM articles WHERE MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root');
+----+-------------------------------------+-----------------+
| id | body                                | score           |
+----+-------------------------------------+-----------------+
|  4 | 1. Never run mysqld as root. 2. ... | 1.5055546709332 |
|  6 | When configured properly, MySQL ... |   1.31140957288 |
+----+-------------------------------------+-----------------+
2 rows in set (0.00 sec)
```

Desde a versão 4.1.1, pesquisas full-text suportam expansão de consulta (em particular, sua variante "blind query expansion"). Ela é geralmente útil quando uma frase pesquisada é muito curta, o que normalmente significa que um usuário está confiando em um conhecimento contido, que a pesquisa full-text normalmente perde. Por exemplo, um usuario pesquisanado por "database" podem realmente significar que "MySQL", "Oracle", "DB2", "RDBMS" são todas frases que devem coincidir com "databases" e devem ser encontrados também. Isto é conhecimento contido. Blind query expansion (also known as automatic relevance feedback) works by performing the search twice, where the search phrase for the second search is the original search phrase concatenated with the few top found documents from the first search. Thus, if one of these documents contained the word "databases" and the word "MySQL", then the second search will find the documents that contain the word "MySQL" but not "database". Another example could be searching for books by Georges Simenon about Maigret, when a user is not sure how to spell "Maigret". Then, searching for "Megre and the reluctant witnesses" will find only "Maigret and the Reluctant Witnesses" without query expansion, but all books with the word "Maigret" on the second pass of a search with query expansion. Note: because blind query expansion tends to increase noise significantly, by returning non-relevant documents, it's only meaningful to use when a search phrase is rather short.

O MySQL utiliza um analizados muito simples para separar texto em palavras. Uma "palavra" é uma sequência de caracteres consistindo de letras, digitos, '', e '_'. Qualquer "palavra" presente na lista de palavra de parada ou for muito curta é ignorada. O tamanho padrão mínimo das palavras que serão encontradas pela pesquisa full-text é de quatro caracteres. Isto pode ser alterado como descrito em Seção 6.8.2 [Fulltext Fine-tuning], Página 624.

Toda palavra correta na lista de coleções e na consulta é pesada de acordo com sua significância na consulta ou coleção. Deste modo, uma palavra que está presente em vários documentos terá peso menor (e poderá ter até mesmo um peso zero), já que ele têm um valor semântico baixo nesta coleção particular. Por outro lado, se a palavra é rara, ela receberá um peso alto. O peso das palavras são então combinados para computar a relevância das linhas.

Tal técnica funciona melhor com coleções grandes (de fato, ela é cuidadosamente ajustado deste modo). Para tabelas muito pequenas, a distribuição das palavras não refletem adequadamente seus valores semânticos, e este modelo pode algumas vezes produzir resultados bizarros.

```
mysql> SELECT * FROM articles WHERE MATCH (title,body) AGAINST ('MySQL');
Empty set (0.00 sec)
```

A busca pela palavra `MySQL` não produz resultados no exemplo acima, porque esta palavra está presente em mais da metade das linhas. Como tal, ela é efetivamente tratada como palavra de parada (isto é, uma palavra com valor semântico zero). Este é o comportamento mais desejável — uma consulta de linguagem natural não deve retornar toda segunda linha de uma tabela de 1 GB.

Uma palavra que casa com metade dos registros em uma tabela tem menos chance de encontrar dosumentos relevantes. De fato, é muito mais provável encontrar vários documentos irrelevantes. Todos nós sabemos que isto acontece com muita frequência quando tentamos encontrar alguma coisa na internet com um mecanismo de busca. É com esta razão que estes registros tem sido atribuído com um baixo valor semântico **neste banco de dados particular**.

Na versão 4.0.1, MySQL também pode realizar buscas full-text booleanas usando o modificador `IN BOOLEAN MODE`.

```
mysql> SELECT * FROM articles WHERE MATCH (title,body)
    ->     AGAINST ('+MySQL -YourSQL' IN BOOLEAN MODE);
+----+-------------------------------+-------------------------------------+
| id | title                         | body                                |
+----+-------------------------------+-------------------------------------+
|  1 | MySQL Tutorial                | DBMS stands for DataBase ...        |
|  2 | How To Use MySQL Efficiently | After you went through a ...        |
|  3 | Optimizing MySQL              | In this tutorial we will show ...   |
|  4 | 1001 MySQL Tricks             | 1. Never run mysqld as root. 2. ... |
|  6 | MySQL Security                | When configured properly, MySQL ... |
+----+-------------------------------+-------------------------------------+
```

Esta consulta recupera todos os registros que contenham a palavra `MySQL` (note: o ponto inicial de 50% não é utilizado), mas que **não** contenha a palavra `YourSQL`. Note que a pesquisa em modo booleano não ordena os registros automaticamente em ordem decrescente de relevância. Você pode ver isto no resultado da consulta anterior, onde a linha com a maior relevância (aquela que contém `MySQL` duas vezes) é listada por último, não em primeiro. Um busca full-text booleana também pode funcionar mesmo sem um índice `FULLTEXT`, no entanto ela seria **lenta**.

A busca full-text booleana suporte potencialmente as seguintes operações:

`+` Um sinal de mais precedente indica que esta palavra **deve estar** presente em cada linha retornada.

`-` Um sinal de menos precedente indice que esta palavra **não deve estar** presente em qualquer linha retornada.

Por padrão (quando nem mais nem menos é especificado) a palavra é opcional, mas as linhas que a contém serão avaliadas positivamente. Isto define o comportamento de `MATCH() ... AGAINST()` sem o modificados `IN BOOLEAN MODE`.

`< >` Estes dois operadores são usados para alterar a contribuição de uma palvara no valor de relevância que á tribuído a um registro. O operador `<` reduz a contribuição e o operador `>` a aumenta. Veja o exemplo abaixo.

`( )` Parenteses são usado para agrupar palavras em subexpressões.

`~` Um til precedente atua como um operador de negação, tornando a contribuição da palavra para a relevância da linha ser negativa. Ele é útil para marcar palavras `"ruidosas"`. Linhas com tais palavras terão uma avaliação mais baixa que outras, mas não será excluída, como seria com o operador `-`.

`*` Um asterisco é um operador de truncamento. Diferente dos outros operadores, ele deve ser **inserida ao fim** da palavra, não deve ser precedente.

`"` A frase que é colocada entre aspas duplas `"`, coincidem apenas com linhas que contenha esta frase **literalmente, como foi digitada**.

E aqui estão alguns exeplos:

`apple banana`
encontra linhas que contenha pela menos uma destas palavras.

`+apple +juice`
... ambas as palavras.

`+apple macintosh`
... palavra “apple”, mas avaliada mais alto se também conter “macintosh”.

`+apple -macintosh`
... palavra “apple” mas não “macintosh”.

`+apple +(>turnover <strudel)`
... “apple” e “turnover”, ou “apple” e “strudel” (em qualquer ordem), mas avalia “apple pie” melhor que “apple strudel”.

`apple*` ... “apple”, “apples”, “applesauce”, e “applet”.

`"some words"`
... “some words of wisdom”, mas não “some noise words”.

6.8.1 Restrições Full-text

- Pesquisas full-text são suportadas apenas por tabelas `MyISAM`.
- Pesquisas full-text pode ser usadas com UCS-2 (mas funcionam com UTF-8 a partir do MySQL 4.1.1).
- Todos os parâmetros da função `MATCH()` devem ser colunas da mesma tabela que é parte do mesmo índice `FULLTEXT`, a menos que `MATCH()` esteja `IN BOOLEAN MODE`.

- Todas as colunas no índice `FULLTEXT` devem ter o mesmo conjunto de caracter.
- A lista de coluna `MATCH()` deve casar exatamente a lista de colunas em algum definição de índice `FULLTEXT` para a tabela, a menos que este `MATCH()` seja `IN BOOLEAN MODE`.
- O argumento para `AGAINST()` deve ser uma string constante.

6.8.2 Ajuste Fino de Pesquisas Full-text no MySQL

Infelizmente, pesquisas full-text ainda possui poucos parâmetros de ajuste, embora adicionar alguns seja de grande prioridade no TODO. Se você tiver uma distribuição fonte do MySQL (veja Seção 2.3 [Instalando o fonte], Página 94), você pode exercer maior controle sobre o comportamenteo de pesquisas full-text.

Note que o busca full-text foi cuidadosamente ajustada para a melhor busca efetiva. Mofificar o comportamento padrão irá, na maioria dos casos, apenas tornar os resultados de busca piores. Não alteren o fonte do MySQL a menos que você saiba o que está fazendo!

A descrição das variáveis full-text na lista a seguir devem ser configuradas no servidor na inicialização. Você não pode modificá-los dinamicamente enquanto o servidor estiver em execução.

- O tamanho mínimo de palavras a serem indexadas é definido pela variavel `ft_min_word_len` do MySQL. Veja Seção 4.6.8.4 [`ft_min_word_len`], Página 309.

  (Esta variável só está disponível a partir do MySQL versão 4.0.)

  O valor padrão é quatro caracteres. Altere-o para o valor de sua preferência e reconstrua os seus índices `FULLTEXT`. Por exemplo, se você quiser pesquisar palavras de três caracteres, você pode definir esta variável colocando a seguinte linha no arquivo de opções:

  ```
  [mysqld]
  ft_min_word_len=3
  ```

  Então reinicie o servidor e reconstrua seus índices `FULLTEXT`.
- A lista de palavras de parada pode ser carregada do arquivo especificado pela variável `ft_stopword_file`. Veja Seção 4.6.8.4 [`ft_stopword_file`], Página 309. Reconstrua o seu índice `FULLTEXT` depois de modificar a lista de palavras de parada. (Esta varável só está disponível a partir do MySQL versão 4.0.10 e posterior)
- O ponto inical de 50% é determinado pelo esquema de pesagem particular escolhido. Para disabilitá-lo, altere a seguinte linha em '`myisam/ftdefs.h`':

  ```
  #define GWS_IN_USE GWS_PROB
  ```

  Para:

  ```
  #define GWS_IN_USE GWS_FREQ
  ```

  Então recompile o MySQL. Não há necessidade de reconstruir o índice neste caso. **Note**: fazendo isto você diminui **em muito** a habilidade do MySQL fornecer valores de relevância adequados para a função `MATCH()`. Se você realmente precisa buscar por tais palavras comuns, seria melhor fazê-lo utilizando `IN BOOLEAN MODE`, que não observa o poonto inicial de 50%.
- Algumas vezes o mantedor do mecanismo de busca gostaria de alterar os operadores usados por busca full-text boolanas. Eles são definidos pela variável `ft_boolean_`

`syntax`. Veja Seção 4.6.8.4 [`ft_boolean_syntax`], Página 309. Ainda, esta variável é somente leitura; este valor está definido em '`myisam/ft_static.c`'.

Para mudanças full-text que exigem que você reconstrua seu índice `FULLTEXT`, o modo mais fácil de fazê-lo para uma tabela `MyISAM` é usar a seguinte instrução, a qual reconstroi o arquivo de índice:

```
mysql> REPAIR TABLE nome_tabela QUICK;
```

6.8.3 TODO de Pesquisas Full-text

- Fazer todas as operações com índices `FULLTEXT` **mais rápidas**.
- Operadores de proximidade
- Supporte para "always-index words". Elas poderiam ser quaisquer strings que o usuário quisesse tratar como palavra, os exemplos são "C++", "AS/400", "TCP/IP", etc.
- Suporte a busca full-text em tabelas `MERGE`.
- Suporte a UCS-2.
- Tornar a lista de palavras de parada dependente da linguagem dos dados.
- Stemming (dependente da linguagem dos dados. é claro).
- Pre-analizadores de UDF genéricas fornecidas pelo usuário.
- Tornar os modelos mais flexíveis (adicionando algum parâmetro ajsutável a `FULLTEXT` em `CREATE/ALTER TABLE`).

6.9 Cache de Consultas do MySQL

A partir da versão 4.0.1, O `servidor MySQL` dispões do recurso `Query Cache` (cache de consultas). Quando em uso, o cache de consultas armazena o textop de uma consulta `SELECT` junto com o resultado correspondente que foi enviado para o cliente. Se uma consulta identica é recebida mais tarde, o servidor retornará o resultado da cache de consultas ao invés de analisar e executar a mesma consulta novamente.

NOTE: A cache de consulta não retornam dados antigos. Quando o dado é modificado, qualquer entrada relevante na cache de consulta é atualizado.

A cache de consultas é extremamente útil em um ambiente onde (algumas) tabelas não mudam com frequência e você tem várias consultas idênticas. Esta é uma situação típica em muitos servidores web que utilizam muito conteúdo dinâmico.

Abaixo está algumas performances de dados da cache de consultas. (Estes resultado foram gerados rodando o pacote de benchmark do MySQL em um Linux Alpha 2 x 500 MHz com 2 GB RAM e uma cache de consultas de 64 MB):

- Se todas as consultas que você estiver realizando forem simples (tais como selecionar um registro de uma tabela com um registro); mas ainda diferente daquelas em que as consultas não são armazendas, a sobrecarga de ter a cache de consultas ativa é de 13%. Este pode ser considerado como o cenário de pior caso. No entanto, na vida real, consultas são muito mais complicadas que nosso exemplo simples, assim a sobrecarga é, normalmente, significantemente menor.

- Buscas depois de uma linha em uma tabela de uma linha é 238% mais rápido. Isto pode ser considerado perto do mínimo de ganho a ser esperado para uma consulta que está armazenada.
- Se você quiser disabilitar o codigo da cache de consulta defina `query_cache_size=0`. Disabilitando o código da cache de consultas não haverá nenhuma sobrecarga notável. (cache de consultas pode ser excluído do código com ajuda da opção de coniguração `--without-query-cache`)

6.9.1 Como a Cache de Consultas Opera

Consultas são comparadas antes da análise, logo

```
SELECT * FROM nome_tabela
```

e

```
Select * from nome_tabela
```

são consideradas consultas diferentes pela cache de consulta, assim consultas precisam ser exatamente a mesma (byte a byte) para serem vistas como idênticas. Além disso, uma consulta pode ser vista como diferente se, por exemplo, um cliente estiver usando um novo formato de protocolo de comunicação ou um conjunto de caracteres diferente de outro cliente.

Cansultas que utilizam banco de dados diferentes, utilizam versões de protocolos diferentes ou que usam conjunto de caracters padrão diferentes são considerados consultas diferentes e armazenadas separadamente.

A cache funciona para consultas do tipo `SELECT SQL_CALC_FOUND_ROWS ...` e `SELECT FOUND_ROWS() ...` porque o número de registros encontrados também é armazenado na cache.

Se o resultado da consulta foi retornado da cache de consultas, então o estado da variável `Com_select` não irá ser aumentado, mas `Qcache_hits` será. Veja Seção 6.9.4 [Estado e Manutenção da Cache de Consultas], Página 628.

Se uma tabela é alterada (`INSERT`, `UPDATE`, `DELETE`, `TRUNCATE`, `ALTER` ou `DROP TABLE|DATABASE`), então todas as caches de consulta que utilizam esta tabela (possivelmente atarvés de uma tabela `MRG_MyISAM`!) se torna inválida e é removida da cache.

Tabelas `InnoDB` transacionais que foram alteradas serão invalidadas quando um `COMMIT` é realizado.

No MySQL 4.0 a cache de consulta está disbilitada dentro da transação (ela não retorna resultados), mas a partir da versão 4.1.1 as caches de consultas funcionarão com tabelas `InnoDB` dentro da transação (ela usará o número da versão da tabela para detectar se a data é atual ou não).

Antes da versão 5.0, consultas com comentários na mesma linha não podem ser trazidas da cache (mas elas serão colocadas na cache se satisfazerem outras condições).

Uma consulta não pode ser armazenada em cache se contem uma das funções:

Função	**Função**	**Função**
`Funções Definidas por Usuarios`	`CONNECTION_ID`	`FOUND_ROWS`

GET_LOCK	RELEASE_LOCK	LOAD_FILE
MASTER_POS_WAIT	NOW	SYSDATE
CURRENT_TIMESTAMP	CURDATE	CURRENT_DATE
CURTIME	CURRENT_TIME	DATABASE
ENCRYPT (com um parâmetro)	LAST_INSERT_ID	RAND
UNIX_TIMESTAMP (sem parâmetros)	USER	BENCHMARK

Um consulta não pode ser armazenada em cache se conter variáveis, referenciar o banco de dados do sistema mysql, for da forma `SELECT ... IN SHARE MODE`, `SELECT ... INTO OUTFILE ...`, `SELECT ... INTO DUMPFILE ...` ou da forma `SELECT * FROM AUTOINCREMENT_FIELD IS NULL` (para retornar a ID da ultima inserção - ODBC contorna este problema).

No entanto, `FOUND_ROWS()` retornará o valor correto, mesmo se a consulta precedente foi buscada da cache.

No caso de uma consulta não utilizar qualquer tabela, ou utilizar tabelas temporárias, ou se o usuário tiver um privilégio de coluna para qualquer tabela chamada, esta consulta não será armazenada em cache.

Antes de uma consulta ser trazida da cache de consulta, o MySQL irá verificar se o usuário com privilégio SELECT para todos os banco de dados e tabelas envolvidos. Se este não for o caso, o resultado em cache não será usado.

6.9.2 Configuração da Cache de Consultas

A cache de consultas adiciona algumas variáveis do sistema `MySQL` para `mysqld` os quais podem ser definidos em um arquivo de configuração, na linha de comando ao iniciar `mysqld`.

- `query_cache_limit` Não armazene em cache resultados que são maiores que isto. (Padrão 1M).
- `query_cache_min_res_unit`

  Esta variável está presente a partir da versão 4.1.

  O resultado de uma consulta (os dados que também são enviados ao cliente) é armazenado na cache de consulta durante o recuperação do resultado. Consequentemente o dado normalmente não é tratado em um grande bloco. A cache de de conaultas aloca blocos para armazenar o dado em demanda, assim quando um bloco é preenchido, um novo bloco é alocado. Como a operação de alocação de memória é caro, a cache de consulta aloca blocos com um tamanho mínimo de `query_cache_min_res_unit`. Quando a consulta é executada, o último bloco do resultado é cortado para o tamanho atual do dado, assim a memória sem uso é liberada.

  - O valor padrão de `query_cache_min_res_unit` é 4 KB o qual deve ser adequada para a maioria dos casos.
  - Se você tiver várias consultas com resultados pequenos, o tamanho padrão do bloco pode levar a fragmentação de memória (indicado por um grande número de blocos livres (`Qcache_free_blocks`), que podem fazer a cache de consultas deletar consultas da cache devido a perda de memória) (`Qcache_lowmem_prunes`)). Neste caso você deve diminuir `query_cache_min_res_unit`.

  - Se você tem muitas consultas com resultados grandes (veja `Qcache_total_blocks` e `Qcache_queries_in_cache`),você pode aumentar a performance aumentadno `query_cache_min_res_unit`. No entanto, seja cuidadoso para não torná-lo muito grande (veja o ponto anterior).
- `query_cache_size` A quantidade de memória (especificada em bytes) alocada para armazenar resultados de consultas antigas. Se ele for 0, a cache de consultas está desbilitada (padrão).
- `query_cache_type` Pode ser atribuido (apenas numérico) com

Opção	Descrição
0	(OFF, não armazene ou retorne resultados)
1	(ON, armazene todos os resultados, exceto consultas `SELECT SQL_NO_CACHE ...`)
2	(DEMAND, armazene apenas cconsultas `SELECT SQL_CACHE ...`)

Dentro de uma thread (conexão), o comportamento da cache de consulta pode ser alterado do padrão. A sintaxe é a seguinte:

```
QUERY_CACHE_TYPE = OFF | ON | DEMAND QUERY_CACHE_TYPE = 0 | 1 | 2
```

Opção	Descrição
0 or OFF	Não armazene ou recupere resultados
1 or ON	Aramazene todos os resultados exceto consultas `SELECT SQL_NO_CACHE ...`.
2 or DEMAND	Armazene apenas consultas `SELECT SQL_CACHE ...`.

6.9.3 Opções da Cache de Consultas na SELECT

Existem duas possibilidades de parâmetros relacionados a cache de consultas que podem ser especificados em uma consulta `SELECT`:

Opção	Descrição
`SQL_CACHE`	Se `QUERY_CACHE_TYPE` é `DEMAND`, permite que a query seja armazenada em cache. Se `QUERY_CACHE_TYPE` é `ON`, este é o padrão. Se `QUERY_CACHE_TYPE` é `OFF`, não faz nada.
`SQL_NO_CACHE`	Faz esta consulta não armazenável em cache, não permite que esta consulta seja armazenada em cache.

6.9.4 Estado e Manutenção da Cache de Consultas

Com o comando `FLUSH QUERY CACHE` você pode desfragmentar a cache de consultas para melhor utilizar a memória. Este comnado não removerá qualquer consulta da cache. `FLUSH TABLES` também descarrega a cache de consultas.

O camnado `RESET QUERY CACHE` remove todas os resultados de consultas da cache de consultas.

Você pode verificar se a cache de consltas está presente em sua versão do MySQL:

```
mysql> SHOW VARIABLES LIKE 'have_query_cache';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
```

```
| have_query_cache | YES   |
+------------------+-------+
1 row in set (0.00 sec)
```

Você pode monitorar o desempenho da cache de consultas com `SHOW STATUS`:

Variável	Descrição
`Qcache_queries_in_cache`	Número de consultas registrada na cache.
`Qcache_inserts`	Número de consultas adicionadas na cache.
`Qcache_hits`	Número de acertos da cache.
`Qcache_lowmem_prunes`	Número de consultas que foram deletadas da cache devido a memória baixa.
`Qcache_not_cached`	N;úmero de consultas não armazenadas em cache (não armazenáveis, ou devido a `QUERY_CACHE_TYPE`).
`Qcache_free_memory`	Quantidade de memória livre para cache de consultas.
`Qcache_free_blocks`	Número de blocos de memória livre na cache de consultas
`Qcache_total_blocks`	Número total de blocos na cache de consultas.

Número total de consultas = `Qcache_inserts + Qcache_hits + Qcache_not_cached`.

A cache de consultas utiliza blocos de tamanhos variáveis, assim `Qcache_total_blocks` e `Qcache_free_blocks` podem indicar fragmentação de memória da cache de consultas. Depois de um `FLUSH QUERY CACHE` apenas um único (grande) bloco livre permanece.

Nota: Toda consulta precisa de um mínimo de 2 blocos (um para o texto da consulta e um ou mais para o resultado da conulta). Também, cada tabela que é usada por uma consulta precisa de um bloco, mas se duas ou mais consultas usam a mesma tabela, apenas um bloco precisa ser alocado.

Você pode utilizar a variável de estado `Qcache_lowmem_prunes` para ajustar o tamanho da cache de consultas. Ela conta o número de consultas que são removidas da cache para liberar memória para armazenar novas consultas. A cache de consultas utiliza uma estratégia `least recently used` (`LRU`) para decidir quais consultas serão removidas da cache.

7 Tipos de Tabela do MySQL

No MySQL Versão 3.23.6, você pode escolher entre 3 formatos de tabelas básicos (`ISAM`, `HEAP` e `MyISAM`). Versões mais novas do MySQL suportam tipos de tabelas adicionais (`InnoDB` ou `BDB`), dependendo de como você o compila. Um banco de dados pode conter tabelas de diferentes tipos.

Ao criar uma nova tabela, você pode dizer ao MySQL que tipo de tabela criar. O tipo de tabela padrão é, normalmente, `MyISAM`.

MySQL sempre criará um arquivo '`.frm`' para guardar as definições de coluna e tabela. Os índices e dados da tabela serão armazenados em um ou mais arquivos, dependendo do tipo de tabela.

Se você tentar utilziar um tipo de tabela que não está ativa ou não foi compilada com o MySQL, ele irá criar uma tabela do tipo `MyISAM`. Este comportamento é conveniente quando você quer copiar tabelas entre servidores MySQL que suportam tipos de tabel;as diferentes. (Talvez o seu servidor master suporte mecanismos de armazenamento tarnsacionais para aumento de segurança, enquanto o servidor slave só utiliza mecanismos de aramazenamento não-transacionais para maior velocidade.)

Esta mudançaautomatica de tipos de tabela podem confuso para novos usuários MySQL. Planejamos arrumar isto introduzindo avisos no protocolo cliente/servidor na versão 4.1 e gerar um aviso quando uma tipo de tabela é automaticamente alterado.

Você pode converter tabelas entre tipos diferentes com a instrução `ALTER TABLE`. Veja Seção 6.5.4 [`ALTER TABLE`], Página 608.

Note que o MySQL suporta dois tipos diferentes de tabelas: tabelas seguras com transação (`InnoDB` and `BDB`) e tabelas não seguras com tarnsação `HEAP`, `ISAM`, `MERGE`, e `MyISAM`).

Vantagens de tabelas seguras com transação (TST):

- Mais segura. Mesmo se o MySQL falhar ou se você tiver problemas com hardware, você pode ter os seus dados de volta, ou através de recuperação automatica ou de um backup + o log de transação.
- Você pode combinar muitas instruções e aceitar todas de uma vez com o comando `COMMIT`.
- Você pode executar um `ROLLBACK` para ignorar suas mudanças (se você não estiver rodando em modo auto-commit).
- Se uma atualização falhar, todas as suas mudanças serão restauradas. (Com tabelas NTST todas as mudanças que tiverem sido feitas são permanentes).
- Pode fornecer melhor concorrência se a tabela obter muitas atualizações concorrentes com leituras.

Note que para utilizar tabelas `InnoDB` você tem que usar pelo menos a opção de inicialização `innodb_data_file_path`. Veja Seção 7.5.3 [InnoDB start], Página 644.

Vantagens de tabelas não seguras com transação (NTST):

- Muito mais rápida e não há nenhuma sobrecarga de transação.
- Usará menos spaço em disco já que não há nenhuma sobrecarga de transação.
- Usará menos memória para as atualizações.

Você pode combinar tabelas TST e NTST na mesma instrução para obter o melhor dos dois mundos.

7.1 Tabelas MyISAM

MyISAM é o tipo de tabela padrão no MySQL Versão 3.23. Ela é baseada no código ISAM e possui várias extensões úteis.

O índice é armazenado em um arquivo com extensão '.MYI' (MYIndex), e os dados são armazenados em um arquivo com a extensão '.MYD' (MYData). Você pode verificar/reparar tabelas MyISAM com o utilitário myisamchk. Veja Seção 4.5.6.7 [Recuperação de Falhas], Página 287. Você pode compactar tabelas MyISAM com myisampack para utilizar menos espaço. Veja Seção 4.8.4 [myisampack], Página 336. O itens seguintes são novos no MyISAM:

- Existe um parâmetro no arquivo MyISAM que indica se a tabela foi fechada corretamente. Se o mysqld é iniciado com --myisam-recover, tabelas MyISAM serão automaticamente verificadas e/ou reparadas na abertura se a tabela não foi fechada apropriadamente.
- Você pode INSERIR novas linhas em uma tabela que não tenha blocos livres no meio do arquivo de dados, na mesma hora outras threadas são lidas da tabela (inserção concorrente). Um bloco livre pode vir de uma atualização de uma linha de tamanho dinâmico com muitos dados para uma linha com menos dados ou ao deletarmos linhas. Quando todos os blocos livres são usados, todas as inserções futurs serão concorrentes de novo.
- Suporte a grandes arquivos (63-bit) em sistema de arquivos/sistemas operacionais que suportam grandes arquivos.
- Todo dado é armazenado com byte mais baixo primeiro. Isto torna a máquina e SO independentes. A única exigência para a portabilidade do arquivo binário é que a a máquina utilize inteiros com sinais em complemento de dois (como toda a máquina nos últimos 20 anos tem) e formato de pontos flutuante IEEE (também totalmente dominante entre máquinas mainstream). A única área de máquinas que não podem suportar compatibilidade binária são sistemas embutidos (porque eles, algumas vezes, tem processadores peculiares).

  Não há uma grande perda de velocidade em armazenar o byte mais baixo de dados primeiro; os bytes em um registro de tabela estão normalmente desalinhados e isto não dá muito poder de leitura do byte desalinhado em outra ordem além da ordem reversa. O código atual busca-valor-coluna também não é crítico em relação ao tempo comparado a outro código.
- Todas as chaves numéricas estão armazendas com o byte mais alto em primeiro para conseguir melhor compactação do índice.
- Tratamento interno de uma coluna AUTO_INCREMENT. MyISAM irá atualizá-lo automaticamenteem um INSERT/UPDATE. O valor AUTO_INCREMENT pode ser zerado com myisamchk. Ele fará colunas AUTO_INCREMENT mais rápidas (pelo menos 10%) e números natigos não irão reutilizar como no antigo ISAM. Note que quando um AUTO_INCREMENT é definido no fim de uma chave multi-parte o comportamento antigo ainda está presente.
- Ao inserir ordenandamente (como quando se utiliza colunas AUTO_INCREMENT) a árvore chave será separada de forma que o nodo mais alto contenha apenas uma chave. Isto irá aumentar a utilização de espaço na árvore de chaves.
- Colunas BLOB e TEXT podem ser indexados.

- Valores `NULL` são perimitidos em colunas indexadas. Isto gasta 0-1 bytes/chave.
- O tamanho máximo da chave é de 500 bytes por padrão (pode ser alterado recomopilando). No caso de chaves maiores que 250 bytes, um tamanho de bloco de chave maior que o padrão de 1024 bytes é usado para esta chave.
- Número máximo de chaves/tabelas é 32 por padrão. Isto pode ser aumentado para 64 sem ser necessário recompilar `myisamchk`.
- `myisamchk` marcará as tabelas como verificadas se alguém executá-las sem `--update-state`. `myisamchk --fast` só verificará aquelas tabelas que não tenham esta marca.
- `myisamchk -a` armazena estatísticas para partes de chaves(e não apenas para toda a chave como no `ISAM`).
- Linhas de tamanho dinâmico serão agora muito menos fragmentados quando misturar deleções com atualizações e inserções. Isto é feito combinando automaticamente blocos deletados adjacentes e extendendo blocos se o próximo bloco é deletado.
- `myisampack` pode empacotar colunas `BLOB` e `VARCHAR`.
- Você pode colocar arquivos de dados e índices em diretórios diferentes para obter maior velocidade (com a opção `DATA/INDEX DIRECTORY="caminho"` para `CREATE TABLE`). Veja Seção 6.5.3 [CREATE TABLE], Página 598.

`MyISAM` também suporta os seguintes itens, os quais o MySQL estará apto a utilizar em um futuro próximo:

- Suporte a tipos `VARCHAR` reais; uma coluna `VARCHAR` inicia com um tamanho armazenado em 2 bytes.
- Tabelas com `VARCHAR` podem ter um registro de tamanho fixo ou dinâmico.
- `VARCHAR` e `CHAR` podem ser maior que 64K. Todos os segmentos de chaves têm a sua própria definição de linguagem. Isto habilitará o MySQL para ter diferentes definições de linguagens por coluna.
- Um índice computado em hash pode ser usado para `UNIQUE`. Isto lhe permitirá ter `UNIQUE` em qualquer combinação de colunas na tabela. (Você não pode procurar em um em um índice computado `UNIQUE`, de qualquer forma.)

Note que os arquivos de índice são muito menores com `MyISAM` que com `ISAM`. Isto significa que `MyISAM` usará normalmente menos recursos do sistema que `ISAM`, mas precisará de mais tempo de CPU quando inserir dados em um índice compactado.

As seguintes opções para `mysqld` podem ser usadas para alterar o comportamento de tabelas `MyISAM`. Veja Seção 4.6.8.4 [SHOW VARIABLES], Página 309.

Opção	**Descrição**
`--myisam-recover=#`	Recuperação automática de tabelas com falhas.
`-O myisam_sort_buffer_size=#`	Buffer utilizado ao recuperar tabelas.
`--delay-key-write=ALL`	Não desarrega buffers de chaves entre escritas para qualquer tabela MyISAM
`-O myisam_max_extra_sort_file_size=#`	Usada paa ajudar o MySQL a decidir quando utilzar o método lento, mas seguro, de criação de índices de cache de chaves. **Note** este parâmetro é dado em megabytes antes da versão 4.0.3 e em bytes a partir desta versão.

`-O myisam_max_sort_file_size=#` Não utilzia o método rápido de ordenação de índice para criar índices se o arquivo temporário se tornasse maior que o valor dado. **Note** que este parâmetro é dado em megabytes antes da versão 4.0.3 e em bytes a partir desta versão.

`-O bulk_insert_buffer_size=#` Tamanho da arvore cache utilizado na otimização de inserções em bloco. **Note** que este é um limite **por thread**!

A recuperação automática é ativada se você iniciar o `mysqld` com `--myisam-recover=#`. Veja Seção 4.1.1 [Opções de linha de comando], Página 207. Na abertura, é verificado se a tabela está marcada como quebrada ou se a variavel de contagem de abertura para esta tabela não é 0 e você a está executando com `--skip-external-locking`. Se nenhuma das verificações acima forem verdadeiras o seguinte ocorre.

- Verifica-se se a tabela possui erros.
- Se encontrarmos um erro, tente fazer um reparação rápida (com ordenação e sem recriar o arquivo de dados) da tabela.
- Se o reparação falhar devido a um erro no arquivo de dados (por exemplo um erro de chave duplicada), é feita uma nova tentativa, mas desta vez o arquivo de dados é recriado.
- Se a reparação falhar, tente mais uma vez com o antigo método de opção de reparação (escrever linha a linha sem ordenação) o qual deve estar apto a reparar qualquer tipo de erros com pequenas exigências de disco.

Se a recuperação não estiver apta a recuperar todas as linhas de uma instrução completada previamente e você não especificou `FORCE` como uma opção para `myisam-recover`, então a reparação automática abortará com uma mensagem de erro no arquivo de erros:

```
Error: Couldn't repair table: test.g00pages
```

Caso você tenha utilizado a opção `FORCE`, você irá obter um aviso no arquivo de erro:

```
Warning: Found 344 of 354 rows when repairing ./test/g00pages
```

Note que se você executar uma recuperação automática com a opção `BACKUP`, você deve ter um script `cron` que mova automaticamente arquivos com nome como '`tablename-datetime.BAK`' do diretório de banco de dados para uma media de backup.

Veja Seção 4.1.1 [Opções de linha de comando], Página 207.

7.1.1 Espaço Necessário para Chaves

O MySQL pode suportar diversos tipos de índices, mas o tipo normal é ISAM ou MyISAM. Eles utilizam um índice de árvore-B, e você pode calcular aproximadamente o tamanho do arquivo de índice como `(key_length+4)/0.67`, somado sobre todas as chaves. (Isto é para o pior caso, quando todas as chaves são inseridas ordenadamente e nós não temos nenhuma chave compactada.)

Índices string são compactados em espaços. Se a primeira parte do índice é uma string, ele também será compactado em prefixo. Compactação em espaço torna o arquivo de índice menor que o indicado acima se a coluna string tem muitos espaços no fim ou é uma coluna `VARCHAR` não usada em sua totalidade. Compactação de prefixo é usado em chaves que

iniciam com uma string. A Compactação de prefixo ajuda se existirem muitas strings com o prefixo idêntico.

Em tabelas `MyISAM`, você também pode utilizar prefixos em números comprimidos especificando `PACK_KEYS=1` quando você cria a tabela. Isto ajuda quando você tem muitas chaves inteiras que têm prefixo idêntico quando o número é armazenado com o byte mais alto primeiro.

7.1.2 Formatos de Tabelas `MyISAM`

`MyISAM` suporta 3 tipos diferentes de tabelas. Dois deles são escolhidos automaticamente dependendo do tipo das colunas que você está usando. O terceiro, tabelas compactadas, só pode ser criado com a ferramenta `myisampack`.

Quando você cria (`CREATE`) ou altera (`ALTER` uma tabela, você pode, para tabelas que não possuem `BLOB`s, forçar o formato da tabela para `DYNAMIC` ou `FIXED` com a opção de tabela `ROW_FORMAT=#`. No futuro você estará apto a compactar/descompactar tabelas especificando `ROW_FORMAT=compressed | default` para `ALTER TABLE`. Veja Seção 6.5.3 [CREATE TABLE], Página 598.

7.1.2.1 Características de Tabelas Estáticas (Tamanho Fixo)

Este é o formato padrão. É usado quando a tabela não contém colunas `VARCHAR`, `BLOB`, ou `TEXT`.

Este é o formato mais e simples e seguro. É também o mais rápidos dos formatos em disco. A velocidade vem da facilidade de se encontrar dados no disco. Procurar por algo com um índice no formato estático é muito simples. Apenas multiplique o número de linhas pelo seu tamanho.

Também, ao varrermos uma tabela, é muito simples ler um número contante de registros a cada leitura de disco.

A segurança é evidenciada se o seu computador falha ao escrever em um arquivo MyISAM de tamanho fixo, caso no qual o `myisamchk` pode facilemente descobrir onde cada linha começa e termina. Assim, geralmente pode se recuperar todos os registros, exceto os escritos parcialmente. Note que no MySQL todos os índices sempre podem ser reconstruídos.

- Todas as colunas `CHAR`, `NUMERIC`, e `DECIMAL` tem espaços adicionados até o tamanho da coluna.
- É muito rápida.
- Facil de se colocar em cache.
- Fácil de reconstruir depois de uma falha, pois os registros estão localizados em posições fixas.
- Não precisa ser reorganizada (com `myisamchk`) a menos que um grande número de registros sejam deletados e você queira retornar espaço de diaco livre ao sistema operacional.
- Normalmente exige mais espaço de disco que tabelas dinâmicas.

7.1.2.2 Características de Tabelas Dinâmicas

Este formato é usado se a tabela contém colunas `VARCHAR`, `BLOB` ou `TEXT`ou se as tabelas são criadas com `ROW_FORMAT=dynamic`.

Este formato é um pouco mais complexo porque cada linha tem que ter um cabeçalho que diz o seu tamanho. Um registro também pode acabar em mais de um local quando fica maior em uma atualização.

Você pode utilizar `OPTIMIZE tabela` ou `myisamchk` para desfragmentar uma tabela. Se você tiver dados estáticos que você acessa/altera demias na mesma tabela, como alguma coluna `VARCHAR` ou `BLOB`, pode ser uma boa idéia mover as colunas dinâmicas para outra tabela apenas para evitar fragmentação.

- Todas as colunas string são dinâmicas (exceto aquelas com tamanho menor que 4).
- Cada registro é precedido por um mapa de bits indicando quais colunas estão vazias (`''`) para colunas string ou zero para colunas numéricas (Isto é diferente de colunas contendo valores `NULL`). Se uma coluna de string tem um tamanho de zero depois da remoção de espaços extras, ou uma coluna numérica tem um valor de zero, isto é marcado no mapa de bits e não é salvado em disco. Strings não vazias são salvas como um byte de tamanho mais o conteudo da string.
- Geralmente utiliza muito menos espaço de disco que tabelas de tamanho fixo.
- Cada registro utiliza apenas o espeço necessário. Se um registro aumenta, ele é separado em varios pedaços, de acordo com a necessidade. Isto resulta em fragmentação do registro.
- Se você atualiza uma linha com informações que ultrapassam o seu tamanho, a linha será fragmentada. Neste caso, você pode precisar executar `myisamchk -r` de tempos em tempos para obter melhor performance. Use `myisamchk -ei nome_tabela` para algumas estatísticas.
- Não é fácil de recontruí-la após uma falha, pois um registro pode ser fragmentado em muitos pedaços e um link (fragmento) pode ser perdido.
- O tamanho esperado para registros de tamanho dinâmico é:

  ```
  3
  + (número de colunas + 7) / 8
  + (número de colunas char)
  + tamanho empacotado de colunas numéricas
  + tamanho das strings
  + (número de colunas NULL + 7) / 8
  ```

  Existe uma penalidade de 6 bytes para cada link. Um registro dinâmico é ligado sempre que uma atualização causa um aumento do registro. Cada novo link terá pelo menos 20 bytes, assim o próximo aumento estará, provavelemente, no mesmo link. Se não, haverá outro link. Você pode checar quantos links existem com `myisamchk -ed`. Todos os links podem ser removidos com `myisamchk -r`.

7.1.2.3 Características de Tabelas Compactadas

Este é um tipo somente leitura que é gerado com a ferramenta opcional `myisampack` (`pack_isam` para tabelas `ISAM`):

- Todas as distribuições MySQL, mesmo aquelas existentes antes do MySQL se tornar `GPL`, podem ler tabelas que forma compactadas com `myisampack`.
- Tabelas compactadas utilizam muito pouco espaço em disco. Isto minimiza o uso de disco, o que é muito bom quando se utiliza discos lentos (com CD-ROMs).
- Cada registro é compactado separadamente (pouca sobrecarga de acesso). O cabeçalho de um registro é fixo (1-3 bytes) dependendo do maior registro na tabela. Cada coluna é compactada diferentemente. Alguns dos tipos de compactação são:
  - Existe, geralmente, uma tabela Huffman diferente para cada coluna.
  - Compactação de espaço de sufixos.
  - Compactação de espaço de prefixos.
  - Números com valor `0` são armazenados usando 1 bit.
  - Se os valores em uma coluna inteira tem uma faixa pequena, a coluna é armazenada usando o menor tipo possível. Por exemplo, uma coluna `BIGINT` (8 bytes) pode ser armazenada como uma coluna `TINYINT` (1 byte) se todos os valores estão na faixa de `0` a `255`.
  - Se uma coluna tem apenas um pequeno conjunto de valores possíveis, o tipo de coluna é convertido para `ENUM`.
  - Uma coluna pode usar uma combinação das compactações acima.
- Pode tratar registros de tamanho fixo ou dinâmico.
- Pode ser descompactada com `myisamchk`.

7.1.3 Problemas com Tabelas `MyISAM`

O formato do arquivo que o MySQL usa para armazenar dados tem sido testado extensivamente, mas sempre há circunstâncias que podem fazer com que tabelas de banco de dados sejam corrompidas.

7.1.3.1 Tabelas `MyISAM` Corrompidas

Mesmo se o formato MyISAM for muito confiável (todas as alterações na tabela são escritas antes da instrução SQL retornar), você ainda pode ter tabelas corrompidas se algum dos seguintes itens ocorrer:

- O processo `mysqld` ser finalizado no meio de uma escrita.
- Finalização inesperada do computador (por exemplo, se o computador é desligado).
- Um erro de hardware.
- Você estar usando um programa externo (como myisamchk) em uma tabela aberta.
- Um bug de um software no código MySQL ou MyISAM.

Os sintomas típicos de uma tabela corrompida são:

- Você obtem o erro `Incorrect key file for table: '...'. Try to repair it` enquanto seleciona dados da tabela.
- Consultas não encontram linhas em uma tabela ou retornam dados incompletos.

Você pode verificar se uma tabela está ok com o comando `CHECK TABLE`. Veja Seção 4.5.4 [CHECK TABLE], Página 278.

Você pode repara um tabela corrompida com `REPAIR TABLE`. Veja Seção 4.5.5 [REPAIR TABLE], Página 279. Você também pode repará-la, quando o `mysqld` não estiver em execução com o comando `myisamchk`. `sintaxe myisamchk`.

Se a sua tabela estiver muito corrompida você deve tentar encontrar o razão! Veja Seção A.4.1 [Falhas], Página 925.

Neste caso, a coisa mais importante de saber é se a tabela foi corrompida porque o `mysqld` foi finalizado (pode se verificar isto facilmente verificando se há uma linha `restarted mysqld` recente no arquivo de erro do mysql. Se este não é o caso, então você deve tentar fazer um caso de teste disto. Veja Seção E.1.6 [Caso de teste reproduzíveis], Página 1084.

7.1.3.2 O Cliente está usando a tabela ou não a fechou de forma apropriada

Cada arquivo '`.MYI`' do `MyISAM` tem um contador no cabeçalho que pode ser usado para verificar se uma tabela foi fechada apropriadamente.

Se você obteve o seguinte aviso de `CHECK TABLE` ou `myisamchk`:

```
# clients is using or hasn't closed the table properly
```

isto significa que este contador eta fora de sincronia. Insto não significa que a tabela está corrompida, mas significa que você poderia pelo menos fazer uma verificação na tabeal para verificar se está ok.

O contador funciona da seguinte forma:

- A primeira vez que a tabela é atualizada no MySQL, um contador no cabeçalho do arquivo de índice é incrementado.
- O contador não é alterado durante outras alterações.
- Quando a última intância da tabela é fechda (devido a um `FLUSH` ou porque não há espaço na cache de tabelas) o contador é decremetado se a tabela tiver sido atualizada em qualquer ponto.
- Quando você raparar a tabela ou verificá-la e ela estiver ok, o contador é zerado.
- Para evitar problemas com interações com outros processos que podem fazer uma verificação na tabela, o contador não é decrementado no fechamento se ele for 0.

Em outras palavras, o único modo dele ficar fora de sincronia é:

- As tabelas `MyISAM` são copiadas sem um `LOCK` e `FLUSH TABLES`.
- O MySQL ter falhado entre uma atualização e o fechamento final. (Note que a tabela pode ainda estar ok já que o MySQL sempre faz escritas de tudo entre cada instrução.)
- Alguém ter feito um `myisamchk --recover` ou `myisamchk --update-state` em uma tabela que estava em uso por `mysqld`.
- Muitos servidores `mysqld` estrem usando a tabela e um deles tiver feito um `REPAIR` ou `CHECK` da tabela enquanto ela estava em uso por outro servidor. Nesta configuração o `CHECK` é seguro de se fazer (mesmo se você obter ovisos de outros servidor), mas `REPAIR` deve ser evitado pois ele atualmente substitui o arquivo de dados por um novo, o qual não é mostrado para os outros servidores.

7.2 Tabelas MERGE

Tabelas MERGE são novas no MySQL Versão 3.23.25. O código ainda está em gamma, mas deve estar razoavelmente estável.

Uma tabela MERGE (também conhecida como tabela MRG_MyISAM) é uma coleção de tabelas MyISAM idênticas que podem ser usada como uma. Você só pode fazer SELECT, DELETE, e UPDATE da coleção de tabelas. Se você fizer um DROP na tabela MERGE, você só está apagando a especificação de MERGE.

Note que DELETE FROM tabela_merge usado sem um WHERE só limpará o mapeamento a tabela, não deletando tudo nas tabeals mapeadas. (Planejamos consertar isto na versão 4.1).

Com tabelas idênticas queremos dizer que todas as tabelas são criadas com informações de colunas e chaves idênticas. Você não pode fundir tabelas nas quais as colunas são empacotadas de forma diferente, não tenham as mesmas colunas ou tenham as chaves em ordem diferente. No entanto, algumas das tabelas podem ser compactadas com myisampack. Veja Seção 4.8.4 [myisampack], Página 336.

Ao criar uma tabela MERGE, você obterá uma arquivo de definição de tabela '.frm' e um arquivo de lista de tabela '.MRG'. O arquivo '.MRG' contém apenas a lista de arquivos índices (arquivos '.MYI') que devem ser usados como um. Antes da versão 4.1.1, todas as tabelas usadas devem estar no mesmo banco de dados assim como a própria tabela MERGE.

Atualmente você precisa ter os privilégios SELECT, UPDATE e DELETE em tabelas mapeadas para uma tabela MERGE.

Tabelas MERGE podem ajudá-lo a resolver os seguintes problemas:

- Facilidade de gernciamento de um conjunto de log de tabelas. Por exemplo, você pode colocar dados de meses diferentes em arquivos separadosfrom different months into separate files, compress some of them with myisampack, and then create a MERGE to use these as one.
- Lhe da maior velocidade. Você pode separar uma grande tabela somente leitura baseado em algum critério e então colocar as diferentes partes da tabela em discos diferentes. Uma tabela MERGE desta forma pode ser muito mais rápida que se usada em uma grande tabela. (Você pode, é claro, usar também um nível RAID para obter o memo tipo de benefício.)
- Faz pesquisas mais eficientes. Se você sabe exatamente o que você esta procurando, você pode buscar em apenas um dos pedaços da tabelas para algumas pesquisas e utilizar tabelas MERGE para outras. Você pode até ter diferentes tabelas MERGE ativas, com possíveis arquivos sobrepostos.
- Reparações mais eficientes. É facil reparar os arquivos individuais que são mapeados para um arquivo MERGE que tentar reparar um arquivo realmente grande.
- Mapeamento instantâneo de diversos arquivos como um. Uma tabela MERGE usa o índice de tabelas individuais. Não é necessário manter um índice de para ela. Isto torna a coleção de tabelas MERGE MUITO rápido de fazer ou remapear. Note que você deve especificar a definição de chave quando você cria uma tabela MERGE!.
- Se você tem um conjunto de tabelas que você junta a uma tabela grande por demanda ou bacth, você deveria criar uma tabela MERGE delas por demanda. Isto é muito mais rápido é economizará bastante espaço em disco.

- Contornam o limite de tamanho de arquivos do sistema operacional.
- Você pode criar um apelido/sinônimo para uma tabela usando `MERGE` sobre uma tabela. Não deve haver nenhum impacto notável na performance ao se fazer isto (apenas algumas chamadas indiretas e chamadas de `memcpy()` para cada leitura).

As desvantagens de tabelas `MERGE` são:

- Você só pode utilizar tabelas `MyISAM` idênticas em uma tabela `MERGE`.
- `REPLACE` não funciona.
- Tabelas `MERGE` usam mais descritores de arquivos. Se você estiver usando uma tabela `MERGE` que mapeia mais de 10 tabelas e 10 usuários a estão usando, você está usando 10*10 + 10 descritores de arquivos. (10 arquivos de dados para 10 usuários e 10 arquivos de índices compartilhados).
- A leitura de chaves é lenta. Quando você faz uma leitura sobre uma chave, o mecanismo de armazenamento `MERGE` precisará fazer uma leitura em todas as tabelas para verificar qual casa melhor com a chave dada. Se você então fizer uma "leia próximo", o mecanismo de armazenamento `MERGE` precisará procurar os buffers de leitura para encontrar a próxima chave. Apenas quando um buffer de chaves é usado, o mecanismo de armazenamento precisará ler o próximo bloco de chaves. Isto torna as chaves `MERGE` mais lentas em pesquisas `eq_ref`, mas não em pesquisas `ref`. Veja Seção 5.2.1 [EXPLAIN], Página 424.
- Você não pode fazer `DROP TABLE`, `ALTER TABLE`, `DELETE FROM nome_tabela` sem uma cláusula `WHERE`, `REPAIR TABLE`, `TRUNCATE TABLE`, `OPTIMIZE TABLE`, ou `ANALYZE TABLE` em nenhuma das tabelas que é mapeada por uma tabela `MERGE` que está "aberta". Se você fizer isto, a tabela `MERGE` pode ainda se referir a tabela original e você obterá resultados inexperados. O modo mais fácil de contornar esta deficiência e através do comando `FLUSH TABLES`, assegurando que nenhuma tabela `MERGE` permanecerá "aberta".

Quando você cria uma tabela `MERGE` você deve especificar com `UNION=(lista-de-tabelas)` quais tabelas você quer usar com uma. Opcionalmente você pode especificar com `INSERT_METHOD` se você quer que inserções em tabelas `MERGE` ocorram na primeira ou na última tabela da lista `UNION`. Se você não especificar `INSERT_METHOD` ou especificar `NO`, entaão todos os comandos `INSERT` na tabela `MERGE` retornarão um erro.

O seguinte exemplo lhe mostra como utilizaqr tabelas `MERGE`:

```
CREATE TABLE t1 (a INT NOT NULL AUTO_INCREMENT PRIMARY KEY, message CHAR(20));
CREATE TABLE t2 (a INT NOT NULL AUTO_INCREMENT PRIMARY KEY, message CHAR(20));
INSERT INTO t1 (message) VALUES ("Testing"),("table"),("t1");
INSERT INTO t2 (message) VALUES ("Testing"),("table"),("t2");
CREATE TABLE total (a INT NOT NULL AUTO_INCREMENT PRIMARY KEY, message CHAR(20))
            TYPE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
SELECT * FROM total;
```

Note que não criamos uma chave `UNIQUE` ou `PRIMARY KEY` na tabela `total` já que a chave não será única na tabela `total`.

Note que você também pode manipular o arquivo '`.MRG`' diretamente de fora do servidor MySQL:

```
shell> cd /mysql-data-directory/current-database
```

```
shell> ls -1 t1.MYI t2.MYI > total.MRG
shell> mysqladmin flush-tables
```

Agora você pode fazer coisas como:

```
mysql> SELECT * FROM total;
+---+---------+
| a | message |
+---+---------+
| 1 | Testing |
| 2 | table   |
| 3 | t1      |
| 1 | Testing |
| 2 | table   |
| 3 | t2      |
+---+---------+
```

Note que a coluna `a`, declarada como `PRIMARY KEY`, não é unica, já que tabelas `MERGE` não podem forca a unicidade sobre um conjunto de tabelas `MyISAM` selecionadas.

Para remapear uma tabela `MERGE` você pode fazer o seguinte:

- Fazer um `DROP` na tabela e recriá-la
- Usar `ALTER TABLE nome_tabela UNION=(...)`
- Alterar o arquivo '`.MRG`' e executar um `FLUSH TABLE` na tabela `MERGE` e todas as tabelas selecionadas para forçar o mecanismo de armazenamento a ler o novo arquivo de definição.

7.2.1 Problemas com Tabelas `MERGE`

Segue abaixo os problemas conhecidos com tabelas `MERGE`:

- Uma tabela `MERGE` não pode manter restrições `UNIQUE` sobre toda tabela. Quando você faz um `INSERT`, os dados vão para a primeira ou última tabela (de acordo com `INSERT_METHOD=xxx`) e estas tabelas `MyISAM` asseguram que os dados são únicos, mas não se sabe nada sobre outras tabelas `MyISAM`.
- `DELETE FROM tabela_merge` usado sem um `WHERE` só limpará o mapeamento da tabela, não deletando tudo na tabela mapeada.
- `RENAME TABLE` em uma tabela usada por uma tabela `MERGE` ativa pode corromper a tabela. Isto será corrigido no MySQL 4.1.x.
- Criação de uma tabela do tipo `MERGE` não verifica se o tabelas selecionadas são de tipos compatíveis ou se elas existem. O MySQL fará uma verificação rápida de se o tamanho do registro é igual entre tabelas mapeadas quando a tabela `MERGE` é usada, mas esta não é uma verificação total.

  Se você usar tabelas `MERGE` deste modo, você poderá obter problemas estranhos.
- Se você usar `ALTER TABLE` para adicionar primeiro um índice `UNIQUE` em uma tabela usada em uma tabela `MERGE` e então usar `ALTER TABLE` para adicionar um índice normal na tabela `MERGE`, a ordem da chave será diferente para as atabelas se houvesse uma chave não única antiga na tabela. Isto ocorre porque `ALTER TABLE` coloca chaves `UNIQUE`

antes de chaves normais para estar apto a detectar chaves duplicadas o mais rápido possível.

- `DROP TABLE` em uma tabela que está em uso por uma tabela `MERGE` não funcionará no Windows porque o mecanismo de armazenamento `MERGE` faz o mapeamento da tabela escondido da camada mais alta do MySQL. Como o Windows não permite que você apague arquivos que estejam abertos, você deve primeiro descarregar todas as tabelas `MERGE` (com `FLUSH TABLES`) ou apagar a tabela `MERGE` antes de apagar a tabela. Nós consertaremos isto assim que introduzirmos `VIEW`s.

7.3 Tabelas `ISAM`

O tipo de tabela `ISAM`, obsoleto, desaparecerá na versão 5.0. Ele está incluído no fonte do MySQL 4.1 é mas não é mais compilado. `MyISAM` é uma implementação melhor deste handler de tabela e você deve converter todas as tabelas `ISAM` para tabelas `MySAM` o mais rápido possível.

`ISAM` usa um índice `B-tree`. O índice é armazenado em um arquivo com a extensão '`.ISM`', e os dados são armazenados em um arquivo com a extensão '`.ISD`'. Você pode verificar/reparar tabelas `ISAM` com o utilitário `isamchk`. Veja Seção 4.5.6.7 [Recuperação de falhas], Página 287.

`ISAM` tem os seguintes recursos/propriedades:

- Chaves compactadas e de tamanho fixo.
- Registros de tamanho fixo e dinâmico
- 16 chaves com 16 chaves parciais/chaves
- Tamanho máximo da chave de 256 (padrão)
- Os dados são armazenados em formato de máquina; isto é rápido mas é dependente da maquina/SO.

A maioria das coisas que são verdadeiras para tabelas `MyISAM` também são verdadeiras para tabelas `ISAM`. Veja Seção 7.1 [Tabelas `MyISAM`], Página 631. As maiores diferenças comparados a tabelas `MyISAM` são:

- Tabelas `ISAM` não são bninários portáveis entre SO/Pataformas.
- Não pode lidar com tabelas > 4G.
- Só suporta compactação de prefixo em strings.
- Limite de chaves menor.
- Tabelas dinâmicas são mais fragmentadas.
- Tableas são compactadas com `pack_isam` ao invés de `myisampack`.

Se você quiser converter uma tabela `ISAM` em uma tabela `MyISAM` de forma a se poder utilizar utilitários tais como `mysqlcheck`, use uma instrução `ALTER TABLE`:

```
mysql> ALTER TABLE nome_tabela TYPE = MYISAM;
```

A versões embutidas do MySQL não supoortam tabelas `ISAM`.

7.4 Tabelas HEAP

Tabeals HEAP usam índices hash e são armazenadas na memória. Isto as torna muito rápidas, mas se o MySQL falhar você irá perder todos os dados armazenados nela. HEAP é muito útil para tabelas temporárias!

As tabelas HEAP do MySQL utilizam hashing 100% dinâmico sem áreas em excesso. Não há espaços extras necessários para listas livres. Tabelas HEAP também não têm problemas com deleção + inserção, o que normalmente é comum em tabelas com hash:

```
mysql> CREATE TABLE test TYPE=HEAP SELECT ip,SUM(downloads) AS down
    ->                   FROM log_table GROUP BY ip;
mysql> SELECT COUNT(ip),AVG(down) FROM test;
mysql> DROP TABLE test;
```

Aqui seguem algumas coisas que você deve considerar ao utilizar tabelas HEAP:

- Você sempre deve utilizar a especificação MAX_ROWS na instrução CREATE para assegurar que você não irá utilizar toda a memória acidentalmente.
- Índices só serão utilizados com = e <=> (mas é MUITO rápido).
- Tabelas HEAP só podem usar chaves inteiras para procurar por uma linha; compare isto a tabelas MyISAM onde qualquer prefixo de chave pode ser usada para encontrar linhas.
- Tabelas HEAP usam um formato de registro de tamanho fixo.
- HEAP não suporta colunas BLOB/TEXT.
- HEAP não suporta colunas AUTO_INCREMENT.
- Antes do MySQL 4.0.2, HEAP não suportava um índice em uma coluna NULL.
- Você pode ter chaves não únicas em uma tabela HEAP (isto não é comum em tabelas com hash).
- Tabelas HEAP são compartilhadas entre todos os clientes (como qualquer outra tabela).
- Você não pode pesquisar pela próxima entrada na ordem (isto é, usar o índice para fazer um ORDER BY).
- Dados de tabelas HEAP são alocados em blocos menores. As tabelas são 100% dinâmicas (na inserção). Não são necessárias areas excessivas e espaço de chave extra. Linhas deletadas são colocadas em uma lista encadeada e são reutilizadas quando você insere novos dados na tabela.
- Você precisa de memória extra suficiente para todas as tabelas HEAP que você quiser utilizar ao mesmo tempo.
- Para liberar memória, você deve executar DELETE FROM tabela_heap, TRUNCATE tabeala_heap ou DROP TABLE tabela_heap.
- O MySQL não pode descobrir aproximadamente quantas linhas existem entre dois valores (isto é utilizado pela atimizador de escala para decidar qual indice usar). Isto pode afetar algumas consultas se você alterar uma tabela MyISAM para uma tabela HEAP.
- Para assegurar que você não vai cometer nenhum erro acidentalmente, você não pode criar tabelas HEAP maiores que max_heap_table_size.

A memória necessária para uma linha na tabela HEAP é:

```
SUM_OVER_ALL_KEYS(max_length_of_key + sizeof(char*) * 2)
+ ALIGN(length_of_row+1, sizeof(char*))
```

`sizeof(char*)` é 4 em uma máquina de 32 bits e 8 em uma máquina de 64 bits.

7.5 Tabelas InnoDB

7.5.1 Visão Geral de Tabelas InnoDB

O `InnoDB` prove o MySQL com um mecanismo de armazenamento seguro com transações (compatível com `ACID`) com commit, rollback, e recuperação em caso de falhas. InnoDB faz bloqueio a nível de registro e também fornece uma leitura sem bloqueio em `SELECT` em um estilo consistente com Oracle. Estes recursos aumentam a performance e a concorrência de multi usuários. Não há a necessidade de escalonamento de bloqueios em `InnoDB`, pois o bloqueio a nível de registro no InnoDB cabe em um espaço muito pequeno. `InnoDB` é o primeiro gerenciador de armazenamento no MySQL que suportam restrições `FOREIGN KEY`.

`InnoDB` foi desenvolvido para obter o máximo de performance ao processar grande volume de dados. Sua eficiência de CPU provavelmente não é conseguido por nenhum outro mecanismo de banco de dados relacional com base em disco.

`InnoDB` é usado na produção de vários sites com banco de dados grandes e que necessitam de alto desempenho. O famoso site de notícias Slashdot.org utiliza `InnoDB`. Mytrix, Inc. armazena mais de 1 TB de dados em `InnoDB`, em outro site trata uma carga média de 800 inserções/atualizações por segundo em `InnoDB`.

Tecnicamente, `InnoDB` é um banco de dados completo colocado sob o MySQL. `InnoDB` tem sua própria área de buffer para armazenar dados e índices na memória principal. `InnoDB` armazena suas tabelas e índices em um espaco de tabela, o qual pode consistir de vários arquivos (ou partições de disco raw). Isto é diferente, por exemplo de tabelas `MyISAM`, onde cada tabela é armazenada como um arquivo separado. Tabelas `InnoDB` podem ser de qualquer tamanho, mesmo em sistemas operacionais onde o sistema de arquivo é limitado a 2 GB.

Você pode encontrar as últimas informações sobre `InnoDB` em `http://www.innodb.com/`. A versão mais atualizada do manual do `InnoDB` sempre é colocada lá.

`InnoDB` é publicade sob a mesma Licença GNU `GPL`, Versão 2 (de Junho de 1991) que MySQL. Se você distribuir MySQL/InnoDB, e sua aplicação não satisfaz as restrições da licença GPL, você deve comprar uma lincença comercial **MySQL Pro** em `https://order.mysql.com/?sub=pg&pg_no=1`.

7.5.2 InnoDB no MySQL Versão 3.23

A partir do MySQL versão 4.0, `InnoDB` está habilitado por padrão. A seguinte informação só se aplica a série 3.23.

Tabelas InnoDB estão incluídas na distribuição fonte a partir do MySQL 3.23.34a e está ativado no binário MySQL -Max da série 3.23. No Windows os binários -Max estão contidos na distribuição padrão.

Se você tiver feito o download de uma versão binária do MySQL que inclui suporte para InnoDB, simplesmente siga as instruções do manual do MySQL para instalar um vrsão binária do MySQL. Se você já tem o MySQL-3.23 instalado, então o modo mais simples de instalar MySQL -Max é substituir i executável do servidor '`mysqld`' com o executável correspondente na distribuição -Max. MySQL e MySQL -Max diferem apenas no executável do servidor. Veja Seção 2.2.9 [Instalando uma distribuição binária], Página 91. Veja Seção 4.8.5 [`mysqld-max`], Página 343.

Para compilar o MySQL com suoprte a InnoDB, faça o download do MySQL-3.23.34a ou posterior de `http://www.mysql.com/` e configure o MySQL com a opção `--with-innodb`. Veja o manual MySQL sobre como instalar uma distribuição fonte. Veja Seção 2.3 [Instalando uma distribuição fonte], Página 94.

```
cd /caminho/para/fonte/mysql-3.23.37
./configure --with-innodb
```

Para utiliar tabelas InnoDB no MySQL-Max-3.23 você **deve** especificar parâmetros de configuração na seção `[mysqld]` do arquivo de configuração '`my.cnf`', ou no Windows opcionalmente em '`my.ini`'.

No mínimo, na versão 3.23 você deve especificar `innodb_data_file_path` onde você especificar o nome e tamanho dos arquivos de dados. Se você não mencionar `innodb_data_home_dir` em '`my.cnf`' o padrão é criar estes arquivoas no `diretorio_dados` do MySQL. Se você especificar `innodb_data_home_dir` como uma string vazia, então você pode dar caminhos absolutos ao seu arquivo de dados em `innodb_data_file_path`.

O modo mínimo de modificar é de adicionar a seção `[mysqld]` a linha

```
innodb_data_file_path=ibdata:30M
```

mas para obter melhor desempenho é melhor que você especifique as opções como recomendado. Veja Seção 7.5.3 [Inicialização InnoDB], Página 644.

7.5.3 Opções de Inicialização do InnoDB

Para habilitar tabelas `InnoDB` no MySQL versão 3.23, veja Seção 7.5.2 [InnoDB in MySQL 3.23], Página 643.

No MySQL-4.0 não é necessário se fazer nada específico para habilitar tabelas `InnoDB`.

O comportamento padrão no MySQL 4.0 e MySQL 4.1 é criar um arquivo '`ibdata1`' auto-extensível de 10 MB no diretório de dados do MySQL e dois '`ib_logfile`'s de 5MB em '`datadir`'. (No MySQL-4.0.0 e 4.0.1 o arquivo de dados é 64 MB e não é auto-extensível.)

Note: Para obter uma boa performance você **deve** definir explicitamente os parâmetros listados nos seguintes exemplos.

Se você não quiser utilizar tabelas `InnoDB`, você pode adicionar a opção `skip-innodb` ao seu arquivo de oção do MySQL.

A partir das versões 3.23.50 e 4.0.2 `InnoDB` permite que o último arquivo de dados n linha `innodb_data_file_path` seja especificado como **auto-extensível**. A sintaxe de `innodb_data_file_path` é a seguinte:

```
caminhodados:tamanhoespec;caminhodados:tamanhoespec;...
...  ;caminhodados:tamanhoespec[:autoextend[:max:tamanhoespec]]
```

Se você especificar o último arquivo de dados coma a opção autoextend, InnoDB extenderá o último arquivo de dados se ele ficar sem espaço no tablespace. O aumento é de 8 MB a cada vez. Um exemplo:

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:100M:autoextend
```

instrui InnoDB a criar apenas um único arquivo de dados com tamanho inicial de 100 MB e que é extendido em blocos de 8 MB quando o espaço acabar. Se o disco ficar cheio você pode querer adicionar outro arquivo de dados a outro disco, por exemplo. Então você tem que olhar o tamanho de 'ibdata1', arredondar o tamanho para baixo até o múltiplo de 1024 * 1024 bytes (= 1 MB) mais próximo, e especificar o tamanho arredondado de 'ibdata1' explicitamente em innodb_data_file_path. Depois disto você pode adicionar outros arquivos de dados:

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:988M;/disk2/ibdata2:50M:autoextend
```

Tenha cuidado com sistema de arquivos onde o tamanho máximo do arquivo é 2 GB. O InnoDB não está ciente disto. Neste sistemas de arquivos você pode querer especificar o tamanho máximo para o arquivo de dados:

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:100M:autoextend:max:2000M
```

Um exemplo de 'my.cnf' simples. Suponha que você tenha um computador com 128 MB RAM e um disco rígido. Abaixo está o exemplo dos parâmetros de configuração possíveis para 'my.cnf' ou 'my.ini' para o InnoDB. Nós consideramos que você está executando MySQL-Max-3.23.50 ou posterior, our MySQL-4.0.2 ou posterior. Este exemplo serve para a maioria dos usuários, tanto em Unix e Windows, que não querem distribuir arquivos de dados InnoDB e arquivos de log em vários discos. Isto cria um arquivo de dados 'ibdata1' auto-extensível e dois arquivos de log 'ib_logfile0' e 'ib_logfile1' do InnoDB no datadir do MySQL (normalmente '/mysql/data'). O arquivo de log 'ib_arch_log_0000000000' do InnoDB também fica em datadir.

```
[mysqld]
# Você pode escrever outras opções do servidor MySQL aqui
# ...
#                                   Arquivos de dados deve estar aptos
#                                   a guardar os seus dados e índices.
#                                   Esteja certo que você tem espaço
#                                   livre suficiente em disco.
innodb_data_file_path = ibdata1:10M:autoextend
#                                   Defina o tamanho da área de buffer com
#                                   50 - 80 % da meória do seu computador
set-variable = innodb_buffer_pool_size=70M
set-variable = innodb_additional_mem_pool_size=10M
#                                   Defina o tamanho do seu arquivo log
#                                   para 25 % da tamanho da área de buffer
set-variable = innodb_log_file_size=20M
set-variable = innodb_log_buffer_size=8M
#                                   Defina ..flush_log_at_trx_commit
```

```
#                                       com 0 se você puder perder
#                                       algumas das ultimas trnsações
innodb_flush_log_at_trx_commit=1
```

Check that the MySQL server has the rights to create files in `datadir`.

Note que os arquivo de dados devem ser < 2 GB em alguns sistemas de arquivos! O tamanho combinado do arquivos de log devem ser < 4 GB. O tamanho combinado dos arquivos de dados devem ser >= 10 MB.

Quando você criar um banco de dados pela primeira vez, é melhor que você inicie o servidor MySQL do prompt de comando. Então InnoDB irá imprimir a informação sobre a criação do banco de dados na tela e você poderá ver o que está acontecendo. Veja abaixo na próxima seção como a saída na tela se parece. Por exemplo, no Windows você pode iniciar '`mysqld-max.exe`' com:

```
your-path-to-mysqld\mysqld-max --console
```

Onde colocar o '`my.cnf`' ou '`my.ini`' no Windows? As regras para o Windows são o seguinte:

- Apenas o '`my.cnf`' ou '`my.ini`' deve ser criado.
- O arquivo '`my.cnf`' deve ser colocado no diretótio raiz do drive '`C:`'.
- O arquivo '`my.ini`' deve ser colocado no diretório WINDIR, e.g, '`C:\WINDOWS`' ou '`C:\WINNT`'. Você pode usar o comando `SET` do MS-DOS para imprimir o valor de WINDIR.
- Se o seu PC utiliza um carrgador de boot onde o drive '`C:`' não é o drive de boot, então a sua única opçào é usar o arquivo '`my.ini`'.

Onde especificar as opções no Unix? No Unix o '`mysqld`' lê opções dos seguintes arquivos, se eles existirem, na seguinte ordem:

- '`/etc/my.cnf`' Opções globais.
- '`COMPILATION_DATADIR/my.cnf`' Opções específicas do servidor.
- '`defaults-extra-file`' O arquivo especificado com `--defaults-extra-file=...`.
- '`~/.my.cnf`' Opções específicas do usuário

'`COMPILATION_DATADIR`' é o dirertório de dados do MySQL o qual foi especificado como uma opção do `./configure` quando o '`mysqld`' foi compilado. (normalmente '`/usr/local/mysql/data`' para uma instalação binária ou '`/usr/local/var`' para uma instalação fonte).

Se você não estiver certo de onde '`mysqld`' lê o seu '`my.cnf`' ou '`my.ini`', você pode dar o caminho como a primeira opção de linha de comando para o servidor: `mysqld --defaults-file=your_path_to_my_cnf`.

O InnoDB forma o caminho do diretório a um arquivo de dados concatenando textualmente `innodb_data_home_dir` a um nome de arquivo de dados ou caminho em `innodb_data_file_path`, adicionando uma possível barra ou barra invertida entre eles se for necessário. Se a palavra-chave `innodb_data_home_dir` não é mencionada em '`my.cnf`', o padrão para ele é o diretório 'ponto' '`./`' que significa o `datadir` de MySQL.

Um exemplo de '`my.cnf`' avançado. Suponha que você tenha um computador Linux com 2 GB RAM e três disco rígidos de 60 GB (no caminho de diretórios '`/`', '`/dr2`' e '`/dr3`'). Abaixo esta um exemplo de parâmetros de configuração possíveis no arquivo '`my.cnf`' para o InnoDB.

Note que o InnoDB não cria diretórios: você mesmo deve criá-los. Use o comando `mkdir` do Unix ou MS-DOS para criar o diretório base do grupo de dados e de log.

```
[mysqld]
# Você pode escrever outras opções do servidor MySQL aqui
# ...
innodb_data_home_dir =
#                                   Os arquivos de devem estar aptos a
#                                   guardar seus dados e índices
innodb_data_file_path = /ibdata/ibdata1:2000M;/dr2/ibdata/ibdata2:2000M:autoextend
#                                   Defina o tamanho da área de buffer para
#                                   50 - 80 % da memória do seu computador,
#                                   mas esteja certo, no Linux x86, que o
#                                   total de memória usada é < 2 GB
set-variable = innodb_buffer_pool_size=1G
set-variable = innodb_additional_mem_pool_size=20M
innodb_log_group_home_dir = /dr3/iblogs
#                                   .._log_arch_dir deve ser o mesmo
#                                   que .._log_group_home_dir
innodb_log_arch_dir = /dr3/iblogs
set-variable = innodb_log_files_in_group=3
#                                   Defina o  tamanho do arquivo de log
#                                   para cerca de 15% do tamanho da
#                                   área da buffer
set-variable = innodb_log_file_size=150M
set-variable = innodb_log_buffer_size=8M
#                                   Defina ..flush_log_at_trx_commit com
#                                   0 se você puder permitir a perda de
#                                   algumas das ultimas transações.
innodb_flush_log_at_trx_commit=1
set-variable = innodb_lock_wait_timeout=50
#innodb_flush_method=fdatasync
#set-variable = innodb_thread_concurrency=5
```

Note que nós colocamos os dois arquivos de dados em discos diferentes. O InnoDB preencherá o tablespace de tabela formado pelos arquivos de dados de baixo para cima. Em alguns casos ele aumentará o desempenho do banco de dados se todos os dados não forem colocados no mesmo disco físico. Colocar os arquivos de log em discos diferentes dos de dados é geralmente, benéfico para o desempenho. Você pode usar **partições de discos raw** (dispositivos raw) como arquivos de dados. Em alguns Unixs eles aumentam a E/S. Vejam a seção sobre gerenciamento de espaço de arquivos no InnoDB para saber como especificá-los no '`my.cnf`'.

Aviso: no Linux x86 você deve ter cuidado par **não definir um uso de memória muito alto**. glibc permitirá que o área do processo cresça acima da pilha da thread, o que fará com que o seu servidor falhe. Isto é um risco se o valor de

```
innodb_buffer_pool_size + key_buffer +
max_connections * (sort_buffer + read_buffer_size) + max_connections * 2 MB
```

é próximo de 2 GB ou exceda 2 GB. Cada thread usará uma pilha (geralmente 2 MB, mas no binário da MySQL AB é somente 256 KB) e no pior caso usará tmabém `sort_buffer + read_buffer_size` de memória adicional.

Como sintonizar outros parâmetros do servidor '`mysqld`'? Valores comuns que servem para a maioria dos usuários são:

```
skip-locking
set-variable = max_connections=200
set-variable = read_buffer_size=1M
set-variable = sort_buffer=1M
#                                  Defina key_buffer com 5 - 50%
#                                  de sua RAM dependendo de quanto
#                                  você usa tabelas MyISAM, mas
#                                  mantenha key_buffer + tamanho da
#                                  área de buffer do InnoDB < 80% de
#                                  sua RAM
set-variable = key_buffer=...
```

Note que alguns parâmetros são dados usando o formato do parâmetro numérico de '`my.cnf`': `set-variable = innodb... = 123`, outros (parâmetros string e booleanos) com outro formato: `innodb_... = ...` .

O significado dos parâmetros de configuração são os seguintes:

Opção	**Descrição**
`innodb_file_per_table`	Disponível a partir da versão 4.1.1. Esta opção faz com que o InnoDB armazene cada tabela criada em seu próprio arquivo '`.ibd`'. Veja a seção sobre multiplos tablespaces.
`innodb_data_home_dir`	A parte comum do caminho do diretório para todos arquivos de dados InnoDB. Se você não mencionar esta opção em '`my.cnf`', o padrão é o `datadir` do MySQL. Você pde especificá-lo também como uma string vazia, e neste caso você poderá utilizar caminhos de arquivos absolutos em `innodb_data_file_path`.

`innodb_data_file_path`	Caminho para os arquivos de dados individuais e os seus tamanhos. O caminho do diretório completo para cada arquivo de dados é obtido concatenando innodb_data_home_dir ao caminho especificado aqui. O tamanho do arquivo é especificado em megabytes, adicionando o 'M' depois da especificação do tamanho. InnoDB também entende a abreviação 'G', 1 G significa 1024 MB. A partir da versão 3.23.44 você pode definir o tamanho do arquivo maior que 4 GB em sistemas operacionais que seuportam que suportam arquivos grandes. Em alguns sistemas operacionais arquivos devem ser menor que 2 GB. Se você não especificar `innodb_data_file_path`, o comportamento padrão a partir do versão 4.0 é criar um arquivo de dados '`ibdata1`' de 10 MB auto-extensível. A soma do tamanho dos arquivos devem ser menores que 10 MB.
`innodb_mirrored_log_groups`	Número de cópias idênticas de grupos de log mantidos para os banco de dados. Atualmente deve ser definido com 1.
`innodb_log_group_home_dir`	Caminho do diretório de arquivos de log do InnoDB. Se você não mencionar esta opção no '`my.cnf`' o padrão é o `datadir` do MySQL.
`innodb_log_files_in_group`	Número de arquivos de log no grupo de log. O InnoDB escreve nos arquivos de modo circular. O valor recomendado aqui é 2. O valor padrão é 2.
`innodb_log_file_size`	Tamanho de cada arquivo de log em um grupo de logs em megabytes. Faixa de valores sensíveis de 1M a 1/n-th do tamanho do área de buffer especificado abaixo, onde n é o número de arquivos de log no grupo. Quanto maior é o valor, menos atividade de descarga é necessária na área de buffer, economizando E/S de disco. Mas arquivos de log maiores também significa que a recuperação será lenta no caso de falhas. O tamanho combinado do arquivo de log deve ser menor que 4GB em comutadores de 32 bits. O padrão é 5M.
`innodb_log_buffer_size`	O tamanho do buffer que o InnoDB utiliza para escrever o log em aruivos no disco. Faixa de valores sensíveis de 1M a 8M. Um buffer de log grande permite aumentar transações para executarem sem precisar de escrever o log em até se fazer um commit da transação. iAlem disso, se você tiver grande transações, fazer um buffer de log maior economiza E/S de disco.

`innodb_flush_log_at_trx_commit`	Normalmente é atribuido 1, significando que em um commit de uma transação o log é descarregado para o disco e as modificações feitas pela transação se tornam permanentes, sobrevivendo a uma falha no banco de dados. Se você estiver disposto a comprometer esta segrança e está executando transações pequenas, você pode definí-lo com 0 ou 2 para reduzir E/S de discos nos logs. O valor 0 significa que o log só é escrito no arquivo e este é descarregado pro disco aproximadamente uma vez por segundo. O valor 2 significa que o log é escrito no arquivo a cada commit, mas o arquivo de log só é descarregado em disco aproximadamente uam vez por segundo. O valor padrão é 1 a partir do MySQL-4.0.13; antes era 0.
`innodb_log_arch_dir`	O diretório onde arquivos de log totalmente escritos seriam escritos se usarmos arquivamento de log. Atualmente o valor deste parâmetro deve ser definido igual a `innodb_log_group_home_dir`.
`innodb_log_archive`	Atualmente este valor deve ser definido com 0. Como a recuperação ai partir de um backup deve ser feito pelo MySQL usando os seus próprios arquivos de log, não há nenhuma necessidade de se arquivos os arquivos de log do InnoDB.
`innodb_buffer_pool_size`	O tamanho do buffer de memória que o InnoDB usa para armazenar dados e índices de suas tabelas. Quanto maior for este valor, menor será a necessidade de E/S de disco para acessar dados na tabela. Em um servidor de banco de dados dedicado você pode definir este parâmetro até 80% do tamanho da memória física da máquina. Não atribua um valor muito alto, pois a competição da memória física pode causar paginação no sistema operacional.
`innodb_buffer_pool_awe_mem_mb`	Tamanho da área de buffer em Mb, se estiver localizado na memória AWE do Windows 32 bits. Deiponível a partir da versão 4.1.0 e relevante apenas no Windows 32 bits. Se o seu Windows suporta mais 4GB de memória, chamado Address Windowing Extensions, você pode alolcar a área de buffer do InnoDB em uma memória física AWE usando este parâmetro. O maior valor possível para isto é 64000. Se este parâmetro for especificado, então innodb_buffer_pool_size é a janela no espaço de endereço de 32 bits do mysqld onde o InnoDB mapeia aquela memória AWE. Um bom valor para innodb_buffer_pool_size é 500M.

`innodb_additional_mem_pool_size`	Tamanho do pool da memória que o InnoDB utiliza para armazenar informações de dicionário de dados e outras estruturas de dados internas. Um bom valor aqui pode ser 2M, mas quanto mais tabelas você tiver em sua aplicação, mais você precisará alocar aqui. Se o InnoDB ficar sem memória neste pool, ele l começara a alocar memória do sistema operacional e a escrever mensagens de aviso no log de erro do MySQL.
`innodb_file_io_threads`	Número de threads de E/S de arquivos no InnoDB. Normalmente ele deve ser 4, mas no Windows E/S de disco pode se beneficiar de um número maior.
`innodb_lock_wait_timeout`	Tempo limite em segundos que uma transação InnoDB pode esperar por uma trava antes de fazer um roll back. InnodDB detecta automaticamente deadlocks de transações em sua própria tabela bloqueada e faz um roll back da transação. Se você utiliza o comando `LOCK TABLES`, ou outro mecanismo de armazenamento seguro com transações diferente do InnoDB na mesma transação, então um deadlock pode crescer, o que não seria notificado pelo InnoDB. Nestes casos o tempo limite é útil para resolver a situação.
`innodb_flush_method`	(Disponível a partir da versão 3.23.40.) O valor padrão para este parâmetro é `fdatasync`. Outra opção é `O_DSYNC`.
`innodb_force_recovery`	Aviso: esta opção só deve ser definida em uma situação de emergência quando você quiser um dump de suas tabelas em um banco de dados corropido! Os valores possíveis são de 1 - 6. Veja abaixo na seção 'Forçando a recuperação' sobre o significado dos valores. Como uma medida segura o InnoDB previne que um usuário modifique os dados quando esta opção é > 0. Esta opção está disponível a partir da versão 3.23.44.

7.5.4 Criando Tablespaces no InnoDB

Suponha que você instalou o MySQL e editou '`my.cnf`' para que ele contenha os parâmetros de configuração do InnoDB necessários. Antes de iniciar o MySQL você deve verificar se os diretórios que você especificou para os arquivos de dados e de log do InnoDB existem e se você tem direito de acesso a estes diretórios. InnoDB não pode criar diretórios, apenas arquivos. Verifique também se você têm espaço suficiente em disco para or arquivos de dados e de log.

Quando iniciar o MySQL, InnoDB começara criando os seus arquivos de dados e de log. O InnoDB irá imprimir algo como o mostrado a seguir:

```
~/mysqlm/sql > mysqld
InnoDB: The first specified datafile /home/heikki/data/ibdata1
did not exist:
InnoDB: a new database to be created!
```

```
InnoDB: Setting file /home/heikki/data/ibdata1 size to 134217728
InnoDB: Database physically writes the file full: wait...
InnoDB: datafile /home/heikki/data/ibdata2 did not exist:
new to be created
InnoDB: Setting file /home/heikki/data/ibdata2 size to 262144000
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file /home/heikki/data/logs/ib_logfile0 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile0 size to 5242880
InnoDB: Log file /home/heikki/data/logs/ib_logfile1 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile1 size to 5242880
InnoDB: Log file /home/heikki/data/logs/ib_logfile2 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile2 size to 5242880
InnoDB: Started
mysqld: ready for connections
```

Um novo banco de dados InnoDB foi criado. Você pode se conectar ao servidor MySQL com o programa cliente MySQL de costume como `mysql`. Quando você finaliza o servidor MySQL com '`mysqladmin shutdown`', a saída do InnoDB será como a seguinte:

```
010321 18:33:34  mysqld: Normal shutdown
010321 18:33:34  mysqld: Shutdown Complete
InnoDB: Starting shutdown...
InnoDB: Shutdown completed
```

Agora você pode ver os diretórios de arquivos de dados e logs e você verá os arquivos criados. O diretório de log também irá conter um pequeno arquivo chamado '`ib_arch_log_0000000000`'. Este arquivo foi resultado da criação do banco de dados, depois do InnoDB desligar o arquivamento de log. Quando o MySQL for iniciado novamente, a saída será a seguinte:

```
~/mysqlm/sql > mysqld
InnoDB: Started
mysqld: ready for connections
```

7.5.4.1 Se Alguma Coisa Der Errado Na Criação Do Banco de Dados

Se o InnoDB imprmir um erro do sistema operacional em uma operação de arquivo normalmente o problema é um dos seguintes:

- Você não criou os diretórios de dados e de logo do InnoDB.
- '`mysqld`' não tem o direito de criar arquivos neste diretório.
- '`mysqld`' não le o arquivo '`my.cnf`' ou '`my.ini`' corretom e consequentemente não enxerga as opções que você especificou.
- O disco está cheio ou a quota de disco foi excedida.
- Você criou um subdiretório cujo nome é igual ao arquivo de dados que você especificou.

- Existe um erro de sintaxe em `innodb_data_home_dir` ou `innodb_data_file_path`.

Se ocorrer algum erro na criação de banco de dados InnoDB, você deve deletar todos os arquivos criados pelo InnoDB. Isto significa todos os arquivos de dados, de log, o pequeno log arquivado e no caso de você já ter criado algumas tableas InnoDB, delete também os arquivos '`.frm`' correspondentes a estas tabelas do diretório de banco de dados do MySQL. Então você pode tentar criar o banco de dados InnoDB novamente.

7.5.5 Criando Tabelas InnoDB

Suponha que você tenha iniciado o cliente MySQL com o comando `mysql test`. Para criar uma tabela no formato InnoDB você deve especificar `TYPE = InnoDB` no comando SQL de criação da tabela:

```
CREATE TABLE CUSTOMER (A INT, B CHAR (20), INDEX (A)) TYPE = InnoDB;
```

Este comando SQL criará uma tabela e um índice na coluna `A` no tablespace do InnoDB consistindo dos arquivos de dados que você especificou em '`my.cnf`'. Adicionalmente o MySQL criará um arquivo '`CUSTOMER.frm`' no diretório de banco de dados '`test`' do MySQL. Internamente, InnoDB adicionará ao seu próprio diretório de dados uma entrada para tabela `'test/CUSTOMER'`. Assim você pode criar uma tabela de mesmo nome `CUSTOMER` em outro banco de dados do MySQL e os nomes de tabela não irão colidir dentro do InnoDB.

Você pode consultar a quantidade de espaço livre no tablespace do InnoDB utilizabdo o comando de status da tabela do MySQL para qualquer tabela que você criou com `TYPE = InnoDB`. Então a quantidade de espaço livre no tablespace aparecerá na seção de comentário da tabela na saída de `SHOW`. Um exemplo:

```
SHOW TABLE STATUS FROM test LIKE 'CUSTOMER'
```

Note que a estatísticas `SHOW` dada sobre tabelas InnoDB são apenas aproximadas: elas não são usadas na otimização SQL. Tamanho reservado de tabelas e índices em bytes estão acurado.

7.5.5.1 Convertendo Tabelas MyISAM para InnoDB

O InnoDB não tem uma otimização especial para criação de índices separados. Assim não há custo para exportar e importar a tabela e criar índices posteriormente. O modo mais rápido de se alterar uma tabela para InnoDB é fazer as inserções diretamente em uma tabela InnoDB, isto é, use `ALTER TABLE ... TYPE=INNODB`, ou crie uma tabela InnoDB vazia com definições idênticas e insira os registro com `INSERT INTO ... SELECT * FROM ...`.

Para obter um melhor controle sobre o processo de inserção, pode ser bom inserir grandes tabelas em pedaços:

```
INSERT INTO newtable SELECT * FROM oldtable
   WHERE yourkey > something AND yourkey <= somethingelse;
```

Depois de todos os dados serem inseridos você pode renomear as tabelas.

Durante a canversão de tabelas grandes você deve configurar á área de buffer com um tamanho grande para reduzir a E/S de disco. Não deve ser maior que 80% da memória física. Você deve configurar o arquivo de log do InnoDB grande, assim como o buffer de log.

Certifique-se de que você não irá ocupar todo o tablespace: tabelas InnoDB gasta muito mais espaço que tabelas MyISAM. Se um `ALTER TABLE` ficar sem espaço, ele irá iniciar um rollback, que pode levar horas se ele estiver no limite de disco. Para inserções, o InnoDB utiliza o buffer de inserção para fundir registros de índices secundários a índices em grupos. Isto economiza muito a E/S de disco. No rollback tal mecanismo não é usado e o rollback pode demorar 30 vezes mais que a inserção.

No caso de um rollback demorado, se você não tiver dados valiosos e seu banco de dados, é melhor que você mate o processo de banco de dados, delete todos os arquivos de dados e de log do InnoDB e todos os arquivos de tabela '`.frm`' e inicie o seu trabalho de novo, do que esperar que milhões de E/Ss de disoc de complete.

7.5.5.2 Restrições `FOREIGN KEY`

A partir da versão 3.23.43b, o InnoDB disponibiliza restrições de chaves estrangeiras. O InnoDB é o primeiro tipo de tabela da MySQL, que permite definir restrições de chaves estrangeiras para guardar a integridade dos seus dados.

A sintaxe da definição das restriçõess de chaves estrangeiras no InnoDB:

```
[CONSTRAINT [symbol]] FOREIGN KEY (index_col_name, ...)
                    REFERENCES nome_tabela (index_nome_coluna, ...)
                    [ON DELETE {CASCADE | SET NULL | NO ACTION
                                | RESTRICT}]
                    [ON UPDATE {CASCADE | SET NULL | NO ACTION
                                | RESTRICT}]
```

Ambas as tabelas devem ser do tipo InnoDB, **na tabela deve existir um índice onde as colunas de chaves estrangeiras listadas como as PRIMEIRAS colunas** e **na tabela indicada deve haver um índice onde as colunas indicadas são listadas como as PRIMEIRAS colunas e na mesma ordem**. O InnoDB não cria índices automaticamente em chaves estrangeiras para chaves referênciadas: você tem que criá-las explicitamente. Os índices são necessários para verificação de chaves estrangeiras para ser rápido e não exigir a varredura da tabela.

Colunas correspondentes nas chaves estrangeiras e a chave referenciada devem ter tipos de dados internos parecidos dentro do InnoDB para que possam ser comparados sem uma conversão de tipo. O **tamanho e a sinalização de tipos inteiros devem ser o mesmo**. O tamanho do tipos string não precisam ser o mesmo. Se você especificar uma ação `SET NULL`, esteja certo de que você **não declarou as colunas na tabela filha** como `NOT NULL`.

Se o MySQL retornar o erro de número 1005 de uma instrução `CREATE TABLE`, e a string de mensagem de erro se referir ao errno 150, então a criação da tabela falhou porque um restrição de chaves estrangeiras não foi formada corretamente. Similarmente, se uma `ALTER TABLE` falhar e se referir ao errno 150, sgnifica que um definição de chave estrangeira foi formada incorretamente na tabela alterada. A partir da versão 4.0.13, você pode usar `SHOW INNODB STATUS` para ver uma explicação detalhada do ultimo erro de chave estrangeira do InnoDB no servidor.

A partir de versão 3.23.50, InnoDB não verifica restrições de chaves estrangeiras naqueles valores de chaves estrangeiras ou chaves referênciadas que contenham uma coluna NULL.

Um desvio do padrão SQL: se na tabela pai existirem diversos registros têm o mesmo valor de chave referência, então o InnoDB atua na verificação da chave estrangeira como o outro

registro pai como se o mesmo valor de chave não existisse. Por exemplo, se você tiver definido uma restrição de tipo `RESTRICT`, e existir um registro filho com diversos registros pais, o InnoDB não permite a deleção de qualquer um dos registros pais.

A partir da versão 3.23.50, você também pode associar a cláusula `ON DELETE CASCADE` ou `ON DELETE SET NULL` com a restrição de chave estrangeira. Opções correspondentes do `ON UPDATE` estão disponíveis a partir da versão 4.0.8. Se `ON DELETE CASCADE` for especificado, e um registro na tabela pai for deletado, então o InnoDB automaticamente também deleta todos aqueles registros na tabela filha cujos valores de chaves estrangeiras são iguais ao valor da chave referênciada no registro pai Se `ON DELETE SET NULL` for especificado, os registros filhos são automaticamente atualizados e assim as colunas na chave estrangeira são definidas com o valor `NULL` do SQL.

Um desvio dos padrões SQL: se `ON UPDATE CASCADE` ou `ON UPDATE SET NULL` retornam para atualizar a MESMA TABELA que ja tenha sido atualizada durante o processo cascata, ele atua como `RESTRICT`. Isto é para prevenirloops infinitos resultantes de atualizações em cascata. Um `ON DELETE SET NULL` auto referêncial, por outro lado, funciona desde a versão 4.0.13. `ON DELETE CASCADE` auto referêncial já está funcionando.

Um exemplo:

```
CREATE TABLE parent(id INT NOT NULL, PRIMARY KEY (id)) TYPE=INNODB;
CREATE TABLE child(id INT, parent_id INT, INDEX par_ind (parent_id),
            FOREIGN KEY (parent_id) REFERENCES parent(id)
            ON DELETE SET NULL
) TYPE=INNODB;
```

Um exemplo complexo:

```
CREATE TABLE product (category INT NOT NULL, id INT NOT NULL,
                      price DECIMAL,
      PRIMARY KEY(category, id)) TYPE=INNODB;
CREATE TABLE customer (id INT NOT NULL,
      PRIMARY KEY (id)) TYPE=INNODB;
CREATE TABLE product_order (no INT NOT NULL AUTO_INCREMENT,
      product_category INT NOT NULL,
      product_id INT NOT NULL,
      customer_id INT NOT NULL,
      PRIMARY KEY(no),
      INDEX (product_category, product_id),
      FOREIGN KEY (product_category, product_id)
        REFERENCES product(category, id)
        ON UPDATE CASCADE ON DELETE RESTRICT,
      INDEX (customer_id),
      FOREIGN KEY (customer_id)
        REFERENCES customer(id)) TYPE=INNODB;
```

A partir da versão 3.23.50 o InnoDB lhe permite adicionar novas restriçoões de chaves estrangeiras a uma tabela.

```
ALTER TABLE seunomedetabela
ADD [CONSTRAINT [symbol]] FOREIGN KEY (...) REFERENCES anothertablename(...)
[on_delete_and_on_update_actions]
```

Lembre-se de criar os índices necessários primeiro.

A partir da versão 4.0.13, o InnoDB suporta

```
ALTER TABLE suatabela DROP FOREIGN KEY id_chave_estrangeira_gerada_internamente
```

Você tem que usar `SHOW CREATE TABLE` para daterminar as id's de chaves estrangeiras geradas internamente quando você apaga uma chave estrangeira.

Na versão anterior a 3.23.50 do InnoDB, `ALTER TABLE` ou `CREATE INDEX` não devem ser usadas em conexões com tabelas que têm restrições de chaves estrangeiras ou que são referênciadas em restrições de chaves estrangeiras: Qualquer `ALTER TABLE` remove todas as restrições de chaves estrangeiras definidas na tabela. Você não deve utilizar `ALTER TABLE` para tabela referenciadas também, mas utilizar `DROP TABLE` e `CREATE TABLE` para modifcar o esquema. Quando o MySQL faz um `ALTER TABLE` ele pode usar internamente `RENAME TABLE`, e isto irá confundir a restrição de chave estrangeira que se refere a tabela. Uma instrução `CREATE INDEX` é processada no MySQL como um `ALTER TABLE`, e estas restrições também se aplicam a ele.

Ao fazer a verificação de chaves estrangeiras, o InnoDB define o bloqueio a nivel de linhas compartilhadas em registros filhos e pais que ele precisa verificar. O InnoDB verifica a restrição de chaves estrangeiras imediatamente: a verificação não é aplicada no commit da transaçao.

Se você quiser ignorar as restrições de chaves estrangeiras durante, por exemplo um operação `LOAD DATA`, você pode fazer `SET FOREIGN_KEY_CHECKS=0`.

O InnoDB lhe permite apagar qualquer tabela mesmo que ela quebre a restrição de chaves estrangeira que referencia a tabela. Ao apagar um tabela restrição que é definida na instrução create também é apagada.

Se você recriar uma tabela que foi apagada, ela deve ter uma definição de acordo com a restrição de chaves estrangeiras que faz referência a ela. Ela deve ter os nomes e tipos de colunas corretor e deve ter os índices na chave referenciada como indicado acima. Se esta condição não for satisfeita, o MySQL retornará o erro de número 1005 e se refere ao errno 150 na string de mensagem de erro.

A partir da versão 3.23.50 o InnoDB retorna da definição de chave estrangeira de uma tabela quando você chama

```
SHOW CREATE TABLE seunometabela
```

Assim o '`mysqldump`' também produz as difinições de tabelas corretas no arquivo dump e não se esquece das chaves estrangeiras.

Você também pode listar as restrições de chaves estrangeiras de uma tabela `T` com

```
SHOW TABLE STATUS FROM seubancodedados LIKE 'T'
```

As restrições de chaves estrangeiras são listadas no comentário da tabela impresso na saída.

7.5.5.3 Multiplos tablespaces - colocando cada tabela em seu próprio arquivo .ibd

NOTA IMPORTANTE: se você atualizar para o InnoDB-4.1.1 ou posterior, será difícil retornar a versão 4.0 ou 4.1.0! Isto ocorre porque versões anteriores do InnoDB não permitem vários tablespaces. Se você precisar retornar para a versão 4.0, você deverá fazer

um dump das tabelas e recriar todo o tablespace do InnoDB. Se você não tiver criado novas tabelas InnoDB em versões posteriores a 4.1.1, e e precisar retornar a versão anterior rapidamente, você pode fazer um downgrade direto para a versão 4.0.18 do MySQL, ou outra da série 4.0. Antes de fazer o downgrade diretamente para a versão 4.0.xx, você terá que finalizar todas as conexões a versões >= 4.1.1 e deixar o '`mysqld`' to run purge and the insert buffer merge to completion, so that `SHOW INNODB STATUS` shows the Main thread in the state `waiting for server activity`. Then you can shut down '`mysqld`' and start 4.0.18 or later in the 4.0 series. A direct downgrade is not recommended, however, because it is not extensively tested.

Starting from MySQL-4.1.1, you can now store each InnoDB table and its indexes into its own file. This feature is called multiple tablespaces, because then each table is stored into its own tablespace.

You can enable this feature by putting the line

```
innodb_file_per_table
```

in the `[mysqld]` section of '`my.cnf`'. Then InnoDB stores each table into its own file '`tablename.ibd`' in the database directory where the table belongs. This is like MyISAM does, but MyISAM divides the table into a data file '`tablename.MYD`' and the index file '`tablename.MYI`'. For InnoDB, both the data and the indexes are in the '`.ibd`' file.

If you remove the line `innodb_file_per_table` from '`my.cnf`', then InnoDB creates tables inside the '`ibdata`' files again. The old tables you had in the '`ibdata`' files before an upgrade to >= 4.1.1 remain there, they are not converted into '`.ibd`' files.

InnoDB always needs the system tablespace, '`.ibd`' files are not enough. The system tablespace consists of the familiar '`ibdata`' files. InnoDB puts there its internal data dictionary and undo logs.

You CANNOT FREELY MOVE .ibd files around, like you can MyISAM tables. This is because the table definition is stored in the InnoDB system tablespace, and also because InnoDB must preserve the consistency of transaction id's and log sequence numbers.

You can move an '`.ibd`' file and the associated table from a database to another (within the same MySQL/InnoDB installation) with the familiar `RENAME` command:

```
RENAME TABLE olddatabasename.tablename TO newdatabasename.tablename;
```

If you have a clean backup of an '`.ibd`' file taken from the SAME MySQL/InnoDB installation, you can restore it to an InnoDB database with the commands:

```
ALTER TABLE tablename DISCARD TABLESPACE; /* CAUTION: deletes the current .ibd file
<put the backup .ibd file to the proper place>
ALTER TABLE tablename IMPORT TABLESPACE;
```

Clean in this context means:

- There are no uncommitted modifications by transactions in the '`.ibd`' file.
- There are no unmerged insert buffer entries to the '`.ibd`' file.
- Purge has removed all delete-marked index records from the '`.ibd`' file.
- '`mysqld`' has flushed all modified pages of the '`.ibd`' file from the buffer pool to the file.

You can make such a clean backup '`.ibd`' file with the following method.

- Stop all activity from the '`mysqld`' server and commit all transactions.

- Wait that `SHOW INNODB STATUS\G` shows that there are no active transactions in the database, and the main thread of InnoDB is `Waiting for server activity`. Then you can take a copy of the '`.ibd`' file.

Another (non-free) method to make such a clean '`.ibd`' file is to

- Use InnoDB Hot Backup to backup the InnoDB installation.
- Start a second '`mysqld`' server on the backup and let it clean up the '`.ibd`' files in the backup.

It is in the TODO to allow moving clean '`.ibd`' files also to another MySQL/InnoDB installation. That requires resetting of trx id's and log sequence numbers in the '`.ibd`' file.

7.5.6 Adicionando e Removendo Arquivos de Dados e Log do InnoDB

A partir da versão 3.23.50 e 4.0.2 você pode especificar o último arquivo de dados InnoDB com `autoextend`. De forma alternativa, pode se aumentar o seu tablespace especificando um arquivo de dados adicional. Para fazer isto você tem que finalizar o servidor MySQL, edite o arquivo '`my.cnf`' adicionando um novo arquivo de dados no `final` de `innodb_data_file_path`, e entao iniciar o servidor MySQL de novo.

Atualmente você não pode remover um arquivo de dados do InnoDB. Para reduzir o tamanho de seu banco de dados você tem que utilizar '`mysqldump`' para fazer um dump de todas as suas tabelas, criar um novo banco de dados e importar suas tabelas para um novo banco de dados.

Se você quiser alterar o número ou o tamanho do seu arquivo de log InnoDB, você tem que finalizar o MySQL e certificar que ele finalizou sem erros. Copie então o arquivo de log antigo em um local seguro apenas para o caso de algo der errado ao finalizar e você precisar recuperar o banco de dados. Delete os arquivos de log antigo do diretório de arquivos de logm edite o '`my.cnf`' e inicie o MySQL novamente. O InnoDB lhe dirá no inicio que ele está criando novos arquivos de log.

7.5.7 Fazendo Backup e Recuperando um Banco de Dados InnoDB

A chave para um gerenciamento seguro de banco de dados é tirar backups regularmente.

O InnoDB Hot Backup é uma ferramenta de backup online que você pode utilizar pra fazer backup dos seus banco de dados InnoDB enquanto ele está executando. O InnoDB Hot Backup não exige que você finalize o seu banco de dados e não realiza nenhum bloqueio ou cria disturbio no seu processo normal de banco de dados. O InnoDB Hot Backup é uma ferramenta adcional paga e que não está incluída na distribuição padrão do MySQL. Veja o site do InnoDB Hot Backup `http://www.innodb.com/manual.php` para informações detalhadas e telas do produto.

Se você puder desativar o servidor MySQL, então, para fazer um backup de 'binario' do seu banco de dados você deve fazer o seguinte:

- Finalize o seu banco de dados MySQL e certifique-se de que ele finalizou sem erros.
- Copie todos os seus arquivos de dados em um local seguro.

- Copie todos os seus arquivos de log do InnoDB em um local seguro.
- Copie o(s) seu(s) arquivo(s) de configuração '`my.cnf`' em um local seguro.
- Copie todos os arquivos '`.frm`' da suas tabelas InnoDB em um local seguro.

Além de fazer um backup de binário descrito acima, você também deve fazer um dump da sua tabela com '`mysqldump`'. A razão para se fazer isto é que um arquivo binário pode ser corrompido cem você perceber. Dumps de tabelas são armazenados em um arquivo texto legível e muito mais simples que arquivos binários de banco de dados. Ver tabelas corropidas através de arquivos de dump é mais fácil e, como o seu formato é simples, a chance dos dados se corromperem seriamente são bem menores.

Uma boa idéia é fazer dumps ao mesmo tempo que você faz o backup de binário do seu banco de dados. Você tem que fechar todos os bancos de dados nos clientes para ter uma cópia consistente de todas as suas tabelas em seu dump. Então você pode fazer o backup de binário e você terá uma cópia consistente de seu banco de dados em dois formatos.

Para se poder recuperar o seu banco de dados InnoDB através do backup de binário descrito acima, você tem que executar o seu banco de dados MySQL com o sistema de log geral e o arquivamento de log do MySQL ligado. Com sistema de log geral nós queremos dizer o mecanismo de log do servidor MySQL que é independente dos logs do InnoDB.

Para recuperação de falhas do seu processo do servidor MySQL, a única coisa que você deve fazer é reiniciá-lo. InnoDB verificará automaticamente os logs e realizará um roll-forward do banco de dados para o situação atual. O InnoDB fará automaticamente um roll back de transações sem commit existentes no momento da falha. Durante a recuperação, InnoDB irá imprimir algo como o seguinte:

```
~/mysqlm/sql > mysqld
InnoDB: Database was not shut down normally.
InnoDB: Starting recovery from log files...
InnoDB: Starting log scan based on checkpoint at
InnoDB: log sequence number 0 13674004
InnoDB: Doing recovery: scanned up to log sequence number 0 13739520
InnoDB: Doing recovery: scanned up to log sequence number 0 13805056
InnoDB: Doing recovery: scanned up to log sequence number 0 13870592
InnoDB: Doing recovery: scanned up to log sequence number 0 13936128
...
InnoDB: Doing recovery: scanned up to log sequence number 0 20555264
InnoDB: Doing recovery: scanned up to log sequence number 0 20620800
InnoDB: Doing recovery: scanned up to log sequence number 0 20664692
InnoDB: 1 uncommitted transaction(s) which must be rolled back
InnoDB: Starting rollback of uncommitted transactions
InnoDB: Rolling back trx no 16745
InnoDB: Rolling back of trx no 16745 completed
InnoDB: Rollback of uncommitted transactions completed
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Apply batch completed
InnoDB: Started
mysqld: ready for connections
```

Se o seu banco de dados for corrompido ou o seu disco falhar, você terá que fazer recuperações de um backup. no caso de dados corropidos, você deve primeiro encontrar um backup que não está corrompido. A partir de um backup, faça a recuperação a partir do arquivo de logs gerais do MySQL de acordo com a instrução no manual do MySQL.

7.5.7.1 Forçando a recuperação

Se ocorre o corrompimento de uma página do banco de dados, você pode desejar fazer um dump de suas tabelas no banco de dados com `SELECT INTO OUTFILE`, e normalmente a maioria dos dados estará intacto e correto. Mas o corrompimento pode fazer com que `SELECT * FROM table`, ou operações de background do InnoDB falhe ou apresentem avisos, ou até mesmo a recuperação roll-forward do InnoDB falhe. A partir do InnoDB 3.23.44, existe uma opção do '`my.cnf`' com a qual você pode forçar o InnoDB a inicializar, e você também pode prevenir que operações de background sejam executadas, e assim você poderá fazer um dump de suas tabelas. Por exemplo, você pode configurar

```
set-variable = innodb_force_recovery = 4
```

no '`my.cnf`'.

As alternativas para `innodb_force_recovery` estão listadas abaixo. O banco de dados não deve ser usado com estas opções! Como medida de segurança o InnoDB previne um usuário de fazer um `INSERT`, `UPDATE`, ou `DELETE` quando esta opção é `>` 0.

A partir da versão 3.23.53 e 4.0.4, você tem permissão de se fazer um `DROP` ou `CREATE` de uma tabela mesmo se a recuperação forçada está sendo usada. Se você sabe que determinada tabela está causando uma falha no rollback, você pode deletá-la. Você pode usar isto também para para um rollback em execução causado por uma falha importanta ou `ALTER TABLE`. Você pode matar o processo mysqld e usar a opção do '`my.cnf`' `innodb_force_recovery=3` para trazer o seu banco de dados sem o rollback. Apague então a tabela que está causando o rollback.

Um número maior abaixo significa que todas as precauções de números menores estão incluídas. Se você puder fazer um dump de todas as suas tabelas com uma opção de no máximo 4, então você está relativamente seguro que apenas alguns dados em paginas individuais corrompidas são perdidos. A opção 6 é mais dramática, porque páginas de bancos de dados são deixadas e um estado obsoleto, que podem introduzir mais corrompimento em árvores-B e outras estruturas de banco de dados.

- 1 (SRV_FORCE_IGNORE_CORRUPT) deixa o servidor executar mesmo se ele detectar uma página corrompida; tenta fazer `SELECT * FROM table` saltar os índices corrompidos e páginas, o que ajuda ao fazer dump de tabelas;
- 2 (SRV_FORCE_NO_BACKGROUND) evita que a thread principal seja executada: se uma falha ocorresse na remoção, isto seria evitado.
- 3 (SRV_FORCE_NO_TRX_UNDO) não executa rollback de transações depois da recuperação;
- 4 (SRV_FORCE_NO_IBUF_MERGE) também previne operações merge no buffer de inserções: se eles causassem falhar, melhor não fazê-los; não calcula as estatísticas da tabelas;
- 5 (SRV_FORCE_NO_UNDO_LOG_SCAN) não procura por undo logs quando iniciar o banco de dados: InnoDB tratará mesmo transações incompletas como comitadas;

- 6 (SRV_FORCE_NO_LOG_REDO) não faça o roll-forward no log em em conexão com recuperação.

7.5.7.2 Ponto de Verificação

O InnoDB implementa um mecanismo de ponto de verificação chamado fuzzy checkpoint. O InnoDB descarregará páginas de banco de dados modificados da áres de buffer em pequenos grupos. Não há necessidade de descarregar a área de buffer em um único grupo, o que iria, na prática, para o processamento da instrução SQL do usuário por um instante.

Na recuperação de falhas o InnoDB procura por um rotulo de ponto de verificação escrito nos arquivos de log. Ele sabe que todas as modificações no banco de dados anteriores ao rótulo já estão presentes na imagem em disco do banco de dados. O InnoDB varre os arquivos de log a partir do ponto de verificação apicando as modificações registradas no banco de dados.

O InnoDB escreve no arquivo de log de um modo circular. Todas as modificações efetivadas que tornam a pagina de banco de dados na área de buffer diferente das imagens em disco devem estar disponíveis no arquivo de log no caso do InnoDB precisar fazer uma recuperação. Isto significa que quando O InnoDB começa a reutilizar um arquivo de log no modo circular, ele deve estar certo de que imagens em disco da pagina de banco de dados já contém as modificações registradas no arquivo de log que o InnoDM irá utilizar. Em outras palavras, o InnoDB precisa criar um ponto de verificação e geralmente isto envolve descarga de páginas de banco de dados modificados para o disco.

O exposto acima explica o porque que fazer o seu arquivo de log muito maior pode economizar E/S de disco com pontos de verificação. Pode fazer sentido configurar o tamanho do arquivo de log tão grande quanto a àrea de buffer ou mesmo maior. O problema com arquivos de log grandes é que a recuperação de falhas pode ser mais demorada pois haverá mais itens a se aplicar ao banco de dados.

7.5.8 Movendo um Banco de Dados InnoDB para Outra Máquina

No Windows o InnoDB armazena os nomes de banco de dados e tabelas internamente sempre em letras minúsculas. Para mover bancos de dados em um formato binário do Unix para o Windows ou do Windows para o Unix você deve ter todas os nomes de tabelas e banco de dados em letras minúscula. Um modo conveniente de fazer isto é adicionar no Unix a linha

```
set-variable=lower_case_table_names=1
```

na seção `[mysqld]` de seu '`my.cnf`' antes de você iniciar a criação de sua tabela. no Windows o valor 1 é o padrão.

Arquivos de dados e log do InnoDB são binários compatíveis com todas as plataformas se o formato do número de ponto flutuante nas máquinas é o mesmo. Você pode mover um banco de dados InnoDB simplesmente copiando todos os arquivos relevantes, os quais nós já listamos na seção anterior sobre backup do banco de dados. Se o formato de ponto flutuante nas máquinas são diferentes mas você não utiliza tipos de dados `FLOAT` ou `DOUBLE` em suas tabelas então o procedimento é o mesmo; apenas copie os arquivos relevantes. Se

os formatos são diferentes e suas tabelas contenham dados de ponto flutuante, você tem que utilizar '`mysqldump`' e '`mysqlimport`' para mover estas tabelas.

Uma dica de desempenho é desligar o modo auto-commit quando você importa dados em seu banco de dados, assumindo que o seu tablespace tem espaço suficiente para o grande segmento de roolback que a transação de importação ira gerar. Só faça o commit depois de importar toda a tabela ou um segmento de uma tabela.

7.5.9 Modelo Transacional do InnoDB

No modelo transacional do `InnoDB` o objetivo é combinar as melhores propriedades de um banco de dados multi-versioning a um bloqueio de duas fases tradicional. O `InnoDB` faz bloqueio a nivel de registro e executa consultas como leitura consistente sem bloqueio, por padrao, no estilo do Oracle. A tabela travada no InnoDB é armazenada com tanta eficiência em relação ao espaço que a escala de bloqueio não é necessária: normalmente diversos usuários tem permissão para bloquear todos os registros no banco de dados, ou qualquer subconjunto aleatório de regitsros, sem que o `InnoDB` fique sem memória.

No `InnoDB` todas as atividades de usuários acontecem dentro de transações. Se o modo autocommit é usado no MySQL, então cada instrução SQL forma uma única transação. O MySQL sempre inicia uma nova conexão com o modo autocommit ligado.

Se o modo autocommit é desligado com `SET AUTOCOMMIT = 0`, então podemos achar que um usuário sempre tem uma transação aberta. Se for executada uma instrução SQL `COMMIT` ou `ROLLBACK`, a transação atual é finalizada e uma nova é iniciada. Ambas instruções liberarão todas as travas do `InnoDB` que foram definidas durante a transação atual. Um `COMMIT` significa que as alterações feitas na transação atual se tornam permanentes e visíveis a outros usuários. Uma instrução `ROLLBACK`, por outro lado, cancela todas as modificações feitas pela transação corrente.

Se a conexão tem `AUTOCOMMIT = 1`, então o usuário pode ainda relaizar uma transação multi-instrução iniciando-a com `START TRANSACTION` ou `BEGIN` e finalizando-a com `COMMIT` ou `ROLLBACK`.

7.5.9.1 InnoDB e `SET ... TRANSACTION ISOLATION LEVEL ...`

Em termos de níveis de isolamento transacional SQL-92, o padrão `InnoDB` é `REPEATABLE READ`. A partir da versão 4.0.5, `InnoDB` oferece todos os níveis de isolamento transacional diferentes descritos pelo padrão SQL-92. Você pode definir o nível de isolamento padrão para todas as conexões na seção `[mysqld]` do '`my.cnf`':

```
transaction-isolation = {READ-UNCOMMITTED | READ-COMMITTED
                         | REPEATABLE-READ | SERIALIZABLE}
```

Um usuário pode alterar o nível de isolamento de um única seção ou todas as próximas seções com a instrução SQL `SET TRANSACTION`. Sua sintaxe é a sseguinte:

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL
                       {READ UNCOMMITTED | READ COMMITTED
                        | REPEATABLE READ | SERIALIZABLE}
```

Note que não há hífens no nome dos níveis na sintaxe SQL.

O comportamento padrão é definir o nível de isolamento para a próxima transação (não iniciada). Se você especificar a palavra chave `GLOBAL` na instrução acima, ela determinará o nível de isolamento globalmente para todas as novas conexões criadas a partir deste ponto (mas não conexão exitentes). Você precisa do privilégio `SUPER` para fazer isto. Usar a palavra chave `SESSION` difine a transação padrão para todas as transações realizadas futuramente na conexão atual. Qualquer cliente é livre para alterar o nível de isolamento da sessão (mesmo no meio de uma transação), ou o nível de isolamento para a próxima transação.

Você pode consultar o nível de isolamento da transação global ou da sessão com:

```
SELECT @@global.tx_isolation;
SELECT @@tx_isolation;
```

Nos travamentos de registro, `InnoDB` usa o chamado bloqueio de chave seguinte (next-key locking). Isto significa que além dos registros de índices, o `InnoDB` também pode bloquear a "lacuna" antes de um registro de índice para bloquear inserções por outros usuários imediatamente antes do registro de índice. Um bloqueio de chave seguinte significa um bloqueio que trava um registro de índice e a lacuna antes dele. O bloqueio de lacuna significa um bloqueio que só trava a lacuna antes do registro de índice.

Uma descrição detalhada de cada nível de isolamento em `InnoDB`:

- `READ UNCOMMITTED` Também é chamada "dirty read": `SELECT`s sem bloqueio são realizados de forma a não procurar por uma possível versão mais nova de um registro; assim as leituras não são 'consistentes' sob este nível de isolamento; de outra forma este nível funciona como `READ COMMITTED`.
- `READ COMMITTED` Nível de isolamento parecido com o Oracle. Todas as instruções `SELECT ... FOR UPDATE` e `SELECT ... LOCK IN SHARE MODE` só travam o registro de índice, `não` a lacuna antes dele e assim permite livre inserção de novos registros próximo ao registro travado. Mas ainda no tipo de faixa `UPDATE` e `DELETE`, o `InnoDB` deve definir lock da chave seguinte ou da lacuna e bloquear inserções feitas por outros usuários nas lacunas cobertas pela faixa. Istó é necessário já que deve se bloquear "linhas fantasmas" para a replicação e recuperação no MySQL funcionar. Leituras consistentes (**Consistent reads**) comportam como no Oracle: cada leitura consistente, mesmo dentro da mesma transação, configura e lê a sua própria cópia recente.
- `REPEATABLE READ` Este é o nível de isolamento padrão do `InnoDB`. `SELECT ... FOR UPDATE`, `SELECT ... LOCK IN SHARE MODE`, `UPDATE`, e `DELETE` que utilizam um índice único com uma condição de busca única, travam apenas o registro de índice encontrado, e não a lacuna antes dele. De outra forma estas operações empregam travamento de registro seguinte, bloqueando a faixa de índice varrida com trava de chave seguinte ou de lacuna e bloqueando novas inserções feitas por outros usuários. Em leituras consistentes (**consistent reads**) existe uma diferença importante do nível de isolmento anterior: neste nível todas as leituras consistentes dentro da mesma transação lêem o mesma cópia estabelecido pela primeira leitura. Esta conversão significa que se você executa diversas `SELECT`s dentro da mesma transação, elas também são consistentes entre elas.
- `SERIALIZABLE` Este nível é como o anterior, mas todos os `SELECT`s são convertidos implicitamente para `SELECT ... LOCK IN SHARE MODE`.

7.5.9.2 Leitura Consistente sem Lock

Uma leitura consistente significa que o InnoDB utiliza multi-versioning para apresentar a uma consulta uma cópia do banco de dados em um dado momento. O consulta verá as mudanças feitas por aquelas transações que fizeram o commit antes daquele momento e não verá nenhuma mudança feita por transações posteriores ou que fizeram o commit. A exceção a esta regra é que a consulta verá as mudanças feitas pela transação que executar a consulta.

Se você está utilizando o nível de isolamento padrão `REPEATABLE READ`, então todas as leituras consistentes dentro da mesma transação lêem a mesma cópia estabelecida pela primeira leitura naquela transação. Você pode obter uma cópia recente para sua consulta fazendo um commit da transação atual e executando uma nova consulta.

Leituras consistentes é o modo padrão no qual o InnoDB processa instruções `SELECT` em níveis de isolamento `READ COMMITTED` e `REPEATABLE READ`. Uma leitura consistentes não configura nenhuma trava em tabelas que ela acessa e assim outros usuários estão livres para modificar estas tabelas ao mesmo tempo que uma leitura consistente esta sendo feita na tabela.

7.5.9.3 Lock de Leitura `SELECT ... FOR UPDATE` e `SELECT ... LOCK IN SHARE MODE`

Uma leitura consistente não é conveniente em alguma circunstâncias. Suponha que você queira adicionar uma nova linha em sua tabela `CHILD`, e está certo que ela já possui um pai na tabela `PARENT`.

Suponha que você utilize leitura consistente para ler a tabela `PARENT` e certamente veja o pai do filho na tabela. Agora você pode adiciona com segurança o registro filho na tabela `CHILD`? Não, porque pode ter acontecido de outro usuário ter deletado o registro pai da tabela `PARENT`, e você não estar ciente disto.

A solução é realizar o `SELECT` em um modo de travamento, `LOCK IN SHARE MODE`.

```
SELECT * FROM PARENT WHERE NAME = 'Jones' LOCK IN SHARE MODE;
```

Realizar uma leitura em modo compartilhado significa que lemos o dado disponível por último e configuramos travas de leitura nos registros lidos. Se o este dado pertencer a uma transação de outro usuário que ainda não fez commit, esperaremos até que o commit seja realizado. Uma trava em modo compartilhado previne que ocorra atualizações ou deleções de registros já lidos. Depois de vermos que a consulta acima retornou o pai `'Jones'`, podemos com segurança adicionar o seu filho a tabela `CHILD`, e realizar o commit de nossa transação. Este exemplo mostra como implementar integridade referêncial no código de sua aplicação.

Deixe-nos mostrar outro exemplo: temos um compo de contador inteiro em uma tabela `CHILD_CODES` que usamos para atribuir um identificador único para cada filho que adicionamos na tabela `CHILD`. Obviamente, usar uma leitura consistente ou uma leitura em modo compartilhado para ler o valor atual do contador não é uma boa idéia, já que dois usuários do banco de dados podem ver o mesmo valor para o contador e, assim, teríamos um erro de chave duplicada ao adicionarmos os dois filhos com o mesmo identificador para a tabela.

Neste caso existem dois bons modos de se implementar a leitura e o incremento do contador: (1) atualizar o contador primeiro aumentando-o de 1 e só depois disto lê-lo, ou (2) ler o contador primeiro com um modo de bloqueio `FOR UPDATE`, e incrementá-lo depois disto:

```
SELECT COUNTER_FIELD FROM CHILD_CODES FOR UPDATE;
UPDATE CHILD_CODES SET COUNTER_FIELD = COUNTER_FIELD + 1;
```

Um `SELECT ... FOR UPDATE` irá ler o dado disponível por último atribuindo travas exclusivas a cada linha que ele ler. Assim ele atribui uma mesma trava que um `UPDATE` SQL pesquisado atribuiria nos registros.

7.5.9.4 Lock da Chave Seguinte: Evitando Problemas com Fantasmas

Em um lock de registro o InnoDB utiliza um algoritmo chamado trava de chave seguinte. O InnoDB faz o lock de registro, assim quando ele faz uma busca ou varre a tabela, ele atribui travas compartilhadas ou exclusivas nos registros que ele encontra. Assim o bloqueio de registro é mais precisamente chamado lock de registro de índice.

A trava que o InnoDB atribui em registro de índices também afetas as 'lacunas' antes daquele registro de índice. Se um usuário tem uma trava compartilhada ou exclusiva no registro R em um índice, então outro usuário não pode inserir um novo registro de índice imediatamente antes de R na ordem do índice. Este bloqueio de lacunas é feito para prevenir o chamado problema de fantasma. Suponha que eu queira ler e travar todos os filhos com identificador maior que 100 da tabela `CHILD` e atualizar alguns campos nos registros selecionados.

```
SELECT * FROM CHILD WHERE ID > 100 FOR UPDATE;
```

Suponha que exista um índice na tabela `CHILD` na coluna `ID`. Nossa consulta varrerá aquele índice começando do primeiro registro onde `ID` é maior que 100. Agora, se a trava atribuída no registro de índice não travasse inserções feitas nas lacunas, um novo filho poderia ser inserido na tabela. Se agora eu executasse em minha transação

```
SELECT * FROM CHILD WHERE ID > 100 FOR UPDATE;
```

novamente, eu veria um novo filho no resultado que a consulta retorna. Isto é contra o princípio de isolamento das transações: uma transação deve executar sem que os dados que ele estaja lendo sejam alterados durante a transação. Se considerarmos um conjunto de registros como um item de dados, então o novo filho 'fantasma' quebrará o principio do isolamento.

Quando o InnoDB varre um índice ele também pode bloquear a lacuna depois do último registro no índice. Assim como no exemplo anterior: a trava atribuida pelo InnoDB irá previnir que seja feita qualquer inserção na tabela onde `ID` seja maior que 100.

Você pode utilizar trava de chave seguinte para implementar uma verificação de unicidade em sua aplicação: se você ler os seus dados em modo compartilhado e não ver um registro que duplique o que você irá inserir, então você pode inserí-lo com segurança e saber que o trava de chave seguinte atribuida ao registro sucessor ao seu durante a leitura irá previnir que alguém insira um registro que duplique o seu neste intervalo. Assim a trava de chave seguinte permite que você 'bloqueie' a não existência de algo em sua tabela.

7.5.9.5 Locks Definidos por Diferentes Instruções SQL no InnoDB

- SELECT ... FROM ...: esta é uma leitura consistente, lendo uma cópia do banco de dados e não defininfo travas.
- SELECT ... FROM ... LOCK IN SHARE MODE: atribui travas de chave seguinte compratilhadas em todos os regitros de índices que a leitura encontrar.
- SELECT ... FROM ... FOR UPDATE: atribui travas de chave seguinte exclusivas em todos os registros de ínidices que a leitura encontra.
- INSERT INTO ... VALUES (...): atribui uma trava exclusiva em registros inseridos; note que está rava não é uma trava de chave seguinte e não previne que outros usuários insiram nas lacunas antes do registro inserido. Se um erro de chave duplicada ocorrerm, atribua uma trava compartilhada no registro de índice duplicado.
- INSERT INTO T SELECT ... FROM S WHERE ... atribui uma trava exclusiva em cada linha inserida em T. Faz a busca em S como uma leitura consistente, mas configura travas de chave seguinte compartilhada em S se o log do MySQL estiver ligado. O InnoDB tem que atribuir travas neste último caso porque em recuperações roll-forward de um backup, toda instrução SQL tem que ser executada exatamente da mesma forma que foi feito originalmente.
- CREATE TABLE ... SELECT ... realiza o SELECT como uma leitura consistente ou com travas compartilhadas, como no item anterior.
- REPLACE é feita como uma inserção se não houver colisões em uma chave única. De outra forma, uma trava de chave seguinte exclusiva é colocada na linha que deve ser atualizada.
- UPDATE ... SET ... WHERE ...: atribui trava de chave seguinte exclusiva em todos os registros que a busca encontrar.
- DELETE FROM ... WHERE ...:atribui trava de chave seguinte exclusiva em todos os registros que a busca encontrar.
- Se uma restrição FOREIGN KEY é definida na tabela. qualquer inserçao, atualização ou deleção que exige verificação da condição de restrição configura travas de registros compartilhados nos registros que que ele olha na verificação da restrição. Também no caso onde a restrição falha. o InnoDB define estes bloqueios.
- LOCK TABLES ... : atribui trava a tabela. Na implementação a camada MySQL de código atribui este bloqueio. A detecção automatica de deadlocks do InnoDB não pode ser feita onde tais travas de tabelas estão envolvidas: veja a seção seguinte. Também, uma vez que o MySQL sabe sobre bloqueio de registros, é impossível que você obtenha um bloqueio em uma tabela na qual outro usuário tenha bloqueio de registro. Mas isto não coloca a integridade da transação em perigo. Veja Seção 7.5.15 [Restrições InnoDB], Página 678.

7.5.9.6 Detecção de Deadlock e Rollback

O InnoDB detecta automaticamente o deadlock de transações e faz um roll back da(s) transação(ões) para prevenir o deadlockck. A partir da versão 4.0.5, o InnoDB tentará escolher pequenas transações para se fazer roll back. O tamanho de uma transação é

determinado pelo número de linhas que foram inseridas, atualizadas ou deletadas. Antes da versão 4.0.5, `InnoDB` sempre fazia roll back da transação cujo pedido de bloqueio fosse o último a criar o deadlock, isto é, um ciclo no grafo de espera da transação.

O `InnoDB` não pode detectar deadlocks onde uma trava atribuida por uma instrução MySQL `LOCK TABLES` está envolvida ou se uma trava definida em outro mecanismo de banco de dados diferente de `InnoDB` está envolvida. Você tem que resolver estas situações usando `innodb_lock_wait_timeout` configurado em '`my.cnf`'.

Quando o `InnoDB` realiza um rollback completo de uma transação, todos as travas da transação são liberadas. No entanto, se é feito o rollback de apenas uma única instrução SQL como um resultado de um erro, algumas das travass definidas pela instrução podem ser preservadas. Isto ocorre porque o `InnoDB` armazena as travas de registro em um formato onde ele não pode saber qual trava foi definida por qual instrução SQL.

7.5.9.7 Um Exemplo de Como a Leitura Consistente Funciona no `InnoDB`

Suponha que você esteja utilizando o nível de isolamento padrão `REPEATABLE READ`. Quando você executa uma leitura consistente, isto é, uma instrução `SELECT` comum, o `InnoDB` dará a sua transação um ponto no tempo de acordo com o que a sua consulta viu no banco de dados Assim, se a transação B deleta uma linha e faz um commit depois que o ponto no tempo foi atribuido, então você não verá a linha deletada. Inserções e atualização são feitos de forma parecida.

Você pode avançar o seu ponto no tempo fazendo um commit da transação e fazendo outro `SELECT`.

Isto é chamado controle de concorrência multi-version.

```
                    User A                  User B

                SET AUTOCOMMIT=0;       SET AUTOCOMMIT=0;
    time
    |           SELECT * FROM t;
    |           empty set
    |                                     INSERT INTO t VALUES (1, 2);
    |
    v           SELECT * FROM t;
                empty set
                                        COMMIT;

                SELECT * FROM t;
                empty set;

                COMMIT;

                SELECT * FROM t;
                --------------------
                |    1    |    2    |
                --------------------
```

Assima o usuário A vê a linha inserida por B apenas quando B fizer um commit da inserção e A tiver feito um commit de sua própria transação pois assim o ponto no tempo é avançado para depois do commit de B.

Se você deseja ver o estado mais atual do banco de dados, você deve utilizar uma trava de leitura:

```
SELECT * FROM t LOCK IN SHARE MODE;
```

7.5.9.8 Como lidar com deadlocks?

Deadlocks são um problema clássico em banco de dados transacionais, mas eles não são perigosos, a menos que eles sejam tão frequentes que você não possa executar certas transações. Normalmente você tem que escrever suas aplicações de forma que elas sempre estejam preparada a reexecutar uma transação se for feito um roll back por causa de deadlocks.

O `InnoDB` utiliza bloqueio automático de registro. Você pode obter deadlocks mesmo no caso de transações que inserem ou deletam uma única linha. Isto ococrre porque estas operações não são realmente 'atômicas': elas automaticamente atribuem travas aos (possivelmente muitos) registros se índices da linha inserida/deletada.

Você pode lidar com deadlocks e reduzí-lo com os seguintes truques:

- Use `SHOW INNODB STATUS` em versões do MySQL posteriores a 3.23.52 e 4.0.3 para determinar a causa do último deadlock. Isto pode lhe ajudar a sintonizar a sua aplicação a avitar travas.
- Sempre estar preparado para reexecutar uma transação se ela falhar em um deadlock. Deadlocks não sÃo perigosos. Apenas tente de novo.
- Commit sua transações com frequência. Transações pequenas têm menos chaces de colidir.
- Se você estiver utilizando as travas de leitura `SELECT ... FOR UPDATE` ou `... LOCK IN SHARE MODE`, tente usar um nível de isolamente mais baixo `READ COMMITTED`.
- Accesse as suas tabelas e linha em uma ordem fixa. Assim as transações formarão filas ordenadas e não entrarão em deadlock.
- Adicione índices bem escolhidos a sua tabela. Então a suas consultas precisarão varrer menos registros de índice e consequentemente atribuirão menos locks. Use `EXPLAIN SELECT` para fazer o MySQL selecione índices apropriados a sua consulta.
- Use menos locks: se você pode utilizar um `SELECT` para retornar dados de uma copia de banco de dados antiga, não adicione a cláusula `FOR UPDATE` ou `LOCK IN SHARE MODE`. Usar o nível de isolamento `READ COMMITTED` é bom aqui, pois cada leitura consistente dentro da mesma transação lê da sua própria cópia atual.
- Se nada ajudar, serialize suas transações com bloqueio de tabela: `LOCK TABLES t1 WRITE, t2 READ, ... ; [faz algo com tabelas t1 e t2 aqui]; UNLOCK TABLES`. Bloqueio de tabela faz com que suas transações se enfilerem em ordem e deadlocks serão evitados. Note que `LOCK TABLES` inicia implictamente uma transação, assim como o comando `BEGIN`, e `UNLOCK TABLES` finaliza implicitamente uma transação em um `COMMIT`.

- Outra solução para colocar transações em série é criar uma tabela 'semáforo' auxiliar onde exista apenas uma única linha. Cada transação atualiza esta linha antes de acessar outra tabela. Deste modo todas as transações acontecem em série. Note que o algoritmo de detecção automático de deadlock do `InnoDB` também funciona pois a trava de série é uma trava de registro. Na trava de tabela do MySQL nós temos que recorrer ao método do tempo limite para resolver um deadlock.

7.5.10 Dicas de Ajuste de Desempenho

1. Se o aplicativo '`top`' do Unix ou o '`Gerenciado de Tarefas`' do Windows mostrar que percentual de uso da CPU com sua carga de trabalho é menor que 70%, provavelmente sua carga de trabalho está no limite do disco. Talvez você esteja fazendo muitos commits de transações ou a área de buffer é muito pequena. Tornar o buffer maior pode lhe ajudar, mas não o configure com mais de 80% da memória física.
2. Envolva diversas modificações em uma transação. O `InnoDB` deve descarregar o log em disco a cada commit da transação se esta transação fizer modificações no banco de dados. Uma vez que o velocidade de rotação do disco é normalmente 167 revoluções/segundo, o número de commits fica limitado aos mesmos 167/segundo se o disco não enganar o sistema operacional.
3. Se você puder ter perda dos últimos commits feitos em transações, você pode configurar o parâmetro `innodb_flush_log_at_trx_commit` no arquivo '`my.cnf`' com 0. O `InnoDB` tenta descarregar o log uma vez por segundo de qualquer forma, embora a descarga não seja garantida.
4. Torne os seus arquivos de log maiores, tão grande quanto a área de buffer. Quando o `InnoDB` escrever o arquivo de log totalmente, ele terá que escrever o conteúdo modificado da área de buffer no disco em um ponto de verificação. Arquivos de log menores causarão muitos escrita desnecessárias em disco. O ponto negativo em arquivos grandes é que o tempo de recuperação será maior.
5. O buffer de log também deve ser grande, cerca de 8 MB.
6. (Relevante para versão 3.23.39 e acima.) Em algumas versões do Linux e Unix, descarregar arquivos em disco com o comando `fdatasync` do Unix e outros métodos parecido é surpreendentemente lento. O método padrão que o `InnoDB` utiliza é a função `fdatasync`. Se você não estiver satisfeito com o desempenho da escrita do banco de dados, você pode tentar configurar `innodb_flush_method` em '`my.cnf`' com `O_DSYNC`, embora `O_DSYNC` pareça ser mais lento em alguns sistemas.
7. Ao importar dados para o `InnoDB`, esteja certo de que o MySQL não está com `autocommit=1` ligado. Assim cada inserção exige uma descarga de log em disco. Coloque antes da linha de importação de arquivo do SQL

   ```
   SET AUTOCOMMIT=0;
   ```

   e depois dele

   ```
   COMMIT;
   ```

   Se você utilizar a opção '`mysqldump`' `--opt`, você obterá arquivos dump que são mais rápidos de importar também em uma tabela `InnoDB`, mesmo sem colocá-los entre `SET AUTOCOMMIT=0; ... COMMIT;`.

8. Tome ciência dos grandes rollbacks de inserções em massa: o `InnoDB` utiliza o buffer de inserção para economizar E/S de disco em inserções, mas em um rollback correspondente tal mecanismo não é usado. Um rollback no limite de disco pode demorar cerca de 30 vezes mais que a insserção correspondente. Matar o processa de banco de dados não irá ajudar pois o rollback irá reiniciar ao se entrar no banco de dados. O único modo de se livrar de um rollback deste tipo é aumentar a área de buffer de forma que o rollback dependa do limite de CPU e seja executado rápidamente ou deltar todo o banco de dados `InnoDB`.
9. Tome ciência também de outras grandeas operações com limite de disco. Use `DROP TABLE` ou `TRUNCATE` (a partiir do MySQL-4.0) para esvaziar uma tabela, não `DELETE FROM suatabela`.
10. Utilize `INSERT` multi-line para reduzir a sobrecarga de comunicação entre o cliente e o servidro se você precisar inserir muitas linhas:

    ```
    INSERT INTO suatabela VALUES (1, 2), (5, 5);
    ```

    Esta dica é válida para inserções em qualquer tipo de tabela, não apenas no `InnoDB`.

7.5.10.1 `SHOW INNODB STATUS` e o Monitor `InnoDB`

A partir da versão 3.23.41, o `InnoDB` inclui o Monitor InnoDB que imprime informações sobre o estado interno do `InnoDB`. A partir das versões 3.23.52 e 4.0.3 você pode usar o comando SQL `SHOW INNODB STATUS` para trazer a saída do Monitor `InnoDB` padrão para o cliente SQL. os dados são úteis para ajuste do desempenho. Se você estiver usando o cliente SQL interativo '`mysql`', a saída é mais legível se você substituir o ponto e vírgula normalmente usado no final das instruções por `\G`:

```
SHOW INNODB STATUS\G
```

Outro modo de usar os Monitores `InnoDB` é deixá-los gravando dados continuamente na saída padrão do servidor '`mysqld`' (nota: o cliente MySQL não exibirá nada). Ao ser ligado, os Monitores InnoDB exibirá dados um vez a cada 15 segundos. Se você executar '`mysqld`' como um daemon então esta saída é normalmente direcionada para o log '`.err`' no `datadir` do MySQL. Este dado é útil para ajuste do desempenho. No Windows você deve iniciar o `mysqld-max` a partir do Prompt do MSDOS com a opção `--standalone --console` para direcionar a saída para a janela do prompt do MS-DOS.

Existe um `innodb_lock_monitor` separada que imprime a mesma informação que `innodb_monitor` mais informações sobre travas configuradas por cada transação.

A informação impressa inclui dados sobre:

- espera de bloqueios de uma transação,
- espera de semáforo de threads,
- pedido de E/S de arquivos pendentes,
- estatísticas de área de buffer e
- atividade de fusão do buffer de inserção e remoção da thread principal do `InnoDB`.

Você pode iniciar o Monitor InnoDB com o seguinte comando SQL:

```
CREATE TABLE innodb_monitor(a INT) type = innodb;
```

e pará-lo com

```
DROP TABLE innodb_monitor;
```

A sintaxe `CREATE TABLE` é só um modo de passar um comando ao mecanismo `InnoDB` através do analisador SQL do MySQL: a tabela criada não é relevante para o Monitor `InnoDB`. Se você fechar o banco de dados quando o manitor estiver em execução, e você quiser iniciar o monitor novamente, você deve apagar a tabela antes de executar um novo `CREATE TABLE` para iniciar o monitor. A sinstaxe pode alterar em distribuição futuras.

Uma saída padrão do Monitor InnoDB:

```
================================
010809 18:45:06 INNODB MONITOR OUTPUT
================================
--------------------------
LOCKS HELD BY TRANSACTIONS
--------------------------
LOCK INFO:
Number of locks in the record hash table 1294
LOCKS FOR TRANSACTION ID 0 579342744
TABLE LOCK table test/mytable trx id 0 582333343 lock_mode IX

RECORD LOCKS space id 0 page no 12758 n bits 104 table test/mytable index
PRIMARY trx id 0 582333343 lock_mode X
Record lock, heap no 2 PHYSICAL RECORD: n_fields 74; 1-byte offs FALSE;
info bits 0
 0: len 4; hex 0001a801; asc ;; 1: len 6; hex 000022b5b39f; asc ";;
 2: len 7; hex 000002001e03ec; asc ;; 3: len 4; hex 00000001;
...
-------------------------------------------
CURRENT SEMAPHORES RESERVED AND SEMAPHORE WAITS
-------------------------------------------
SYNC INFO:
Sorry, cannot give mutex list info in non-debug version!
Sorry, cannot give rw-lock list info in non-debug version!
-----------------------------------------------------
SYNC ARRAY INFO: reservation count 6041054, signal count 2913432
4a239430 waited for by thread 49627477 op. S-LOCK file NOT KNOWN line 0
Mut ex 0 sp 5530989 r 62038708 sys 2155035;
rws 0 8257574 8025336; rwx 0 1121090 1848344
-----------------------------------------------------
CURRENT PENDING FILE I/O'S
--------------------------
Pending normal aio reads:
Reserved slot, messages 40157658 4a4a40b8
Reserved slot, messages 40157658 4a477e28
...
Reserved slot, messages 40157658 4a4424a8
Reserved slot, messages 40157658 4a39ea38
Total of 36 reserved aio slots
```

```
Pending aio writes:
Total of 0 reserved aio slots
Pending insert buffer aio reads:
Total of 0 reserved aio slots
Pending log writes or reads:
Reserved slot, messages 40158c98 40157f98
Total of 1 reserved aio slots
Pending synchronous reads or writes:
Total of 0 reserved aio slots
-----------
BUFFER POOL
-----------
LRU list length 8034
Free list length 0
Flush list length 999
Buffer pool size in pages 8192
Pending reads 39
Pending writes: LRU 0, flush list 0, single page 0
Pages read 31383918, created 51310, written 2985115
----------------------------
END OF INNODB MONITOR OUTPUT
============================
010809 18:45:22 InnoDB starts purge
010809 18:45:22 InnoDB purged 0 pages
```

Algumas notas sobre a saída:

- Se a seção LOCKS HELD BY TRANSACTIONS relatar espera de bloqueios, então a sua aplicação pode ter diputa de travas. A saida também ajuda a rastrear as razões de deadlocks nas transações.
- A seção SYNC INFO irá relatar semáforos reservados se você compilar o InnoDB com `UNIV_SYNC_DEBUG` definido em '`univ.i`'.
- A seção SYNC ARRAY INFO relatas as threads que esperam por semáforos e estatísticas sobre quantas vezes a thread precisou esperar por um mutex ou por um semáforo de trava de leitura/escrita. Um número grande de espera da thread pelo semáforo pode ser um resultado de E/S de disco ou problemas de disputa dentro do InnoDB. As disoutas pode ser devido a paralelismo pesado de consultas ou problemas na programação das threads no sistema operacional.
- A seção CURRENT PENDING FILE I/O'S lista os pedidos de E/S de arquivos que estão pendente. Um número grande indica que a carga de trabalho esta no limite de disco.
- A seção BUFFER POOL lhe dá estatíticas sobre leitura e escrita das páginas. Você pode calcular a partir destes números quanto de E/S em arquivos de dados a sua consulta esta fazendo atualmente.

7.5.11 Implementação de Multi-versioning

Como o InnoDB é um banco de dados multi-version, ele deve mantar informações de versões antigas de seus registros na tablespace. Esta informação é armazenada na estrutura de dados que chamamos de segmento rollback como uma estrutura de dados anoga no Oracle.

Internamente o InnoDB adiciona dois campos a cada linha armazenada no banco de dados. Um campo de 6 bytes diz ao identificador da transação sobrea a última transação que inseriu ou atualizou um registro. Uma deleção também é tratada internamente como uma atualização ande um bit especial é definido para indicae a daleção. Cada linha contém também um campo de 7 bytes chamado roll pointer. O roll pointer aponta para um registro log de itens a desfazer escrito no segmento rollback. Se o registro foi atualizado, então este registro de log contém a informação necessária para reconstruir o conteúdo da linha antes de ela ter sido atualizada.

O InnoDB usa a informação no segmento rollback para realizar o operação de desfazer necessária em um rollback de uma transação. Ele também usa a informação para construir versões mais novas de um registro para uma leitura consistente.

Os logs de itens a desfazer em um segmwnto rollback são divididos en logs de inserção e atualização. Logs de inserção só são necessários em rollback das transações e podem ser discartados assim que se fizer o commit das transações. Logs de atualização também são utilizados em leituras consistentes, e eles só podem ser descartados quando não houver mais transações para as quais o InnoDB atribuiu uma cópia do banco de dados que precisasse das informações do log de atualizações em uma leitura consistente para construir uma versão mais nova do registro do banco de dados.

Você deve se lembrar de fazer commit em suas transaçãoes regularmente, inclusive aquelas transações que só fazem leituras consistentes. Senão o InnoDB não pode descartar dados do log de atualização e o segmento rollback pode crescer demias, enchendo o seu tablespace.

O tamanho fíisico de um registro log de itens a desfazer em um segmento rollback é normalmente menor que o registro inserido ou atualizado correspondente. Você pode usar esta informação para calcular o espaço necessário para o seu segmento rollback.

Neste esquema multi-versioning uma linha não é fisicamente removida do banco de dados imediatamente quando você a deleta com uma instrução SQL. Apenas quando o InnoDB puder descartar o registro de log de itens a desfazer da atualização ele pode, também, remover fisicamente a linha correspondente e seu registros de índices do banco de dados. Esta operação de remoção é chamada 'purge' e é bem rápida, tendo, normalmente, a mesma ordem de tempo da instrução SQL que fez a deleção.

7.5.12 Estrutura de Tabelas e Índices

O MySQL armazena suas informações de dicionários de dados de tabelas em arquivos '`.frm`' no diretório de banco de dados. Mas todo tabela do tipo InnoDB também tem sua própria entrada no dicionários de dados interno do InnoDB dentro da tablespace. Quando o MySQL apaga uma tabela ou um banco de dados, ele tem que deletar o(s) arquivo(s) '`.frm`' e a entrada correspondente dentro do dicionário de dados do InnoDB. Esta é a razão pela qual você não pode mover tabelas InnoDB entre banco de dados simplesmente movendo

os arquivos '`.frm`' e porque `DROP DATABASE` não funcionava em tabelas do tipo InnoDB em versÕes do MySQL anteriores a 3.23.43.

Toda tabela InnoDB tem um índice especial chamado de índice agrupado onde os dados dos registros são armazenados. Se você definir um chave primaria (`PRIMARY KEY`) na sua tabela, então o índice da chave primária será o índice agrupado.

Se você não definir uma chave primária para a sua tabela, o InnoDB irá gerar internamente um índice agrupado qonde as linhas são ordenadas pela ID da linha que o InnoDB atribui as linhas nestas tabelas. O ID da linha é um campo de 6 bytes que cresce quando novas linhas são inseridas. Assim as linhas armazenadas pela sua ID estarão fisicamente na ordem de inserção.

Acessar uma linha pelo índice agrupado é rápido porque os dados do registro estarão na mesma página que a busca de índice nos indicar. Em muitos bancos de dados, os dados são armazenados em página diferente daquela em que se encontra os registros de índices, Se uma tabela é grande, a arquitetura do índice agrupado geralmente economiza E/S de disco se coparado a solução tradicional.

O registro em índices não agrupados (também os chamamos de índices secundários) em InnoDB contém o valor da chave primária para a linha. O InnoDB usa este valor de chave primária para buscar o registro do índice agrupado. Note que se a chave primária for grande, os índices secundários irão utilizar ainda mais espaço.

7.5.12.1 Estrutura Física do Índice

Todos os índices no InnoDB são árvores-B onde os registros de índice são armazenados na página de folhas da árvore, O tamanho padrão de uma página de índice é 16 Kb. Quando novos registros são inseridos, InnoDB tenta deixar 1 / 16 de paginas livre para futuras inserções e atualzações de registro de índices.

Se registros de índice são inseridos em ordem sequencial (ascendente ou descendente, os páginas de índices resultantes estarão cerce de 15/16 completa. Se os registros são inseridos em ordem aleatoria, então as páginas estarão de 1/2 a 15/16 completos. Se o fator de preenchimento de uma página índice ficar abaixo de 1/2, o InnoDB tentará contrair o árvore de índice para liberar a página.

7.5.12.2 Buffer de Inserção

É uma situação comum em aplicativos de banco de dados que a chave prmária seja um identificador único e os novos registros são inseridos em ordem crescente de acordo com a chave primária. Assim a inserção nos índices agrupados não exigem leituras aleatorias a disco.

Por outro lado, índices secundários são normalmente não são únicos e inserções acontecem em uma ordem relativamente aleatória nos índices secundários. Isto causaria diversos acessos de E/S aleatórios em disco sem um mecanismo especial usado em InnoDB.

Se um registro de índice deve ser inserido a um índice secundário que não é único, o InnoDB verifica se a página de índice secundário já está na área de buffer. Se este for o caso, o InnoDB fará a inserção diretamente ná página do índice. Mas, se a página de índice não for encontrada na área de buffer, O InnoDB insere o registro em uma estrutura de buffer

de inserção especial. O buffer de inserção é mantido tão pequeno que ele cabe totalmente na área de buffer e inserções nele podem ser feitas muito rápido.

O buffer de inserção é unido periodicamente à árvore de índices secundários no banco de dados. Geralmente nós podemos juntar diversas inserções na mesma página na árvore índice o que economiza E/S de disco. Buffers de inserções podem aumentar a velocidade das inserções em uma tabela em cerca de 15 vezes.

7.5.12.3 Índices Hash Adaptativos

Se um banco de dados couber quase totalmente na memória principal, então o modo mais rápido de realizar consultas nela é usar índices hash. O InnoDB tem um mecanismo automatico que monitora as buscas em índices feitas nso índices definidos na tabela e, se o InnoDB notar que as consultas podiam ser beneficiadas da construçã de índices hash, tal índice é automaticamente construído.

Mas note que um índice hash é sempre construído com base em um índice de árvore-B existente na tabela. O InnoDB pode construir um índice hash em um prefixo de qualquer tamanho da chave definida pela árvore-B, dependendo de que padrão de busca o InnoDB observa em índices de árvore-B. Um índice hash pode ser parcial: não é exigido que todo o índice seja armazenado na área de buffer. O InnoDB contruirá índices hash por demanda naquelas páginas de índice que são frequentemente acessadas.

Deste forma, Através do mecanismo de índice hash adptativo o InnoDB se adapta a uma memória principal ampla, aporoximando-se da arquitetura dos bancos de dados de memória principal.

7.5.12.4 Estrutura dos Registros Físicos

- Cada registro de índice no InnoDB contém um cabeçalho de 6 bytes. O cabeçalho é usado para ligar registros consecutivos e também para bloqueio de regiostros.
- Registros em índices agrupados contém capos para todas as colunas definidas definidas pelo usuário. Adicionalmente, existe um campo de 6 bytes para a ID da transação e um campo de 7 bytes para o roll pointer.
- Se o usuário não tiver definido uma chave prmiária para uma tabela, então cada registro de índice agrupado também contém um campo ID de 6 bytes.
- Cada registro de índice secundário também contém todos os campos definidos para a chave de índice agrupado.
- Um registro também contém um ponteiro para cada campo do registro. Se o tamanho total dos campos em um registro é menor que 128 bytes, então o ponteiro é de 1 byte, senão é de 2 bytes.

7.5.12.5 Como Funciona uma Coluna `AUTO_INCREMENT` no InnoDB

Depois que um banco de dados inicia, quando um usuário faz a primeira inserção em uma tabela `T` onde uma coluna auto-increment foi definida, e o usuário não fornece um valor explicito para a coluna, então o InnoDB executa `SELECT MAX(auto-inc-column) FROM T`,

e atribui aquele valor incrementado de um a coluna e ao contador de auto incremento da tabela. Dizemos que o contador de auto incremento para a tabela `T` foi inicializado.

O InnoDB segue o mesmo procedimento na inicialização do contador de auto incremento para uma tabela recem criada.

Note que se o usuário especifica em uma inserção o valor 0 a coluna auto-increment. o InnoDM trata a linha como se o valor não tivesse sido especificado.

Depois do contador de auto incremento tiver sido inicializado, se um usuário insere uma linha onde especificamos explicitamente o valor da coluna e o valor é maior que o valor atual do contador, então o contador é configurado com o valor especificado. Se o usuário não especificar um valor explicitamente, o InnoDB incrementa a contador de um e atribui o seu novo valor a coluna.

O mecanismo de auto incremento, ao atribuir valor ao contador, desvia de manipuladores de travas e transações. De outra forma você também pode obter lacuas na sequência de números se você fizer um roll back da transação que tiver obtido números do contador.

O comportamento do auto incremento não é definido se um usuário passar um valor negativo a coluna ou se o valor se tornar maior que o valor inteiro máximo que pode ser armazenado no tipo inteiro especificado.

7.5.13 Gerenciamento do Espaço de Arquivos e E/S de Disco

7.5.13.1 E/S de Disco

Na E/S de disco o InnoDB usa E/S assíncrona. No Windows NT ele usa a E/S assíncrona nativa fornecida pelo sistema operacional. No Unix, o InnoDB usa E/S assíncrona simulada construída dentro do InnoDB: o InnoDB cria um número de threads de E/S que cuidam das operações de E/S, tais como leitura. Em uma versão futura adcionaremos suporte para E/S simulada no Windows NT e E/S nativa nas versões de Unix que possuam este recurso.

No Windows NT o InnoDB usa E/S sem buffer. Isto significa que as páginas de disco que o InnoDB lê ou escreve não são armazenadas na cache de arquivo do sistema operacional. Isto economiza um pouco da banda de memória.

A partir da versão 3.23.41, o InnoDB usa uma técnica de descarga de arquivo da novel chamado escrita dupla (doublewrite). Ela adiciona segurança a recuperação em falhas depois de uma falha do sistema operacional ou queda de força e aumenta o desempenho na maioria dos sistemas Unix, reduzindo a necessidade de operações fsinc.

Escrita dupla significa que antes do InnoDB escrever páginas em um arquivo de dados, ele primeiro os escreve em área de tablespaces contínuos chamados de buffer de escrita dupla (doublewrite buffer). Apenas após a escrita e a descarga no buffer de escrita dupla tiver sido completada, o InnoDB escreve a página em sua posição apropriada na arquivo de dados. Se o sistema operacional falhar no meio da escrita da página, o InnoDB irá fazer a recuperação procurando uma cópia da página no buffer de escrita dupla.

A partir da versão 3.23.41 você também pode usar uma partição de disco raw como um arquivo de dados, mas insto ainda não foi testado. Quando você cria um navo arquivo de dados você tem que colocar a palavra chave `newraw` imediatamente depois do tamanho

do arquivo de dados em `innodb_data_file_path`. A partição deve ter, pelo menos, o tamanho que você especificou. Note que 1M no InnoDB é 1024 x 1024 bytes, enquanto na especificação de disco 1 MB normalmente significa 1000 000 bytes.

```
innodb_data_file_path=/dev/hdd1:5Gnewraw;/dev/hdd2:2Gnewraw
```

Quando você reinicia o banco de dados você **deve** alterar a palavra chave para `raw`. Senão o InnoDB escreverá sobre a sua partição!

```
innodb_data_file_path=/dev/hdd1:5Graw;/dev/hdd2:2Graw
```

Usando um disco raw você pode ter E/S sem buffer em algumas vesões de Unix.

Quando você usar partições de disco raw, certifique-se de que você tem permissões que permitem acesso de leitura e escrita na conta usada para executar o servidor MySQL.

Existem duas heurísticas read-ahead no InnoDB: read-ahead sequencial e read-ahead aleatória. Na read-ahead sequencial o InnoDB percebe que o padrão de acesso a um segmento no tablespace é sequencial. então o InnoDB enviará uma grupo de leitura das paginas do banco de dados para o sistema de E/S. No read-ahead aleatório o InnoDB percebe que algumas áreas no tablespace parecem estar no processo de serem totalmente lidas na área de buffer. O InnoDB envia as leituras remanescente para o sistema de E/S.

7.5.13.2 Gerenciamento do Espaço de Arquivo

Os arquivos de dados definido no arquivo de configuração forma o tablespace do InnoDB. Os arquivos são simplesmente concatenado para formar o tablespace, não há nenhuma listagem em uso. Atualmente você não pode definir onde suas tabelas serão alocadas no tablespace. No entanto, em um tablespace criado recentemente, o InnoDB alocará espaço a partir do low end

O tablespace consiste de páginas de banco de dados cujo tamanho padrão é 16 KB. As páginas são agrupadas numa extendsão de 64 páginas consecutivas. Os 'arquivos' dentro de um tablespace são chamados segmentos no InnoDB. O Nome do segmento rollback é um tanto enganador porque na verdade ele contém vários segmentos no tablespace.

Para cada índice no InnoDB nós alocamos dois segmentos: um é para nós que não são folhas da árvore-B e outro é para nós de folhas. A idéia aqui é conseguir melhorar a "sequencialidade" dos nós de folhas, que comtêm os dados.

Quando um segmento cresce dentro da tablespace, o InnoDB aloca as primeiras 32 páginas para ele, individualmente. Depois disto o InnoDB inicia a alocação de toda a extensão do segmento. O InnoDB pode adicionar a um grande segmento até 4 extensões de uma vez para assegurar a boa "sequencilidade" dos dados.

Algumas páginas na tablespace contém bitmaps de outras páginas e dessa forma algumas poucas extensões em um tablespace do InnoDB não podem ser alocadas ao segmento como um todo, mas apenas como páginas individuais.

Quando você executa uma consulta `SHOW TABLE STATUS FROM ... LIKE ...` para saber sobre o espaço livre disponível no tablespace, o InnoDB irá relatar as extensões que estejam definitivamente livres na tabelspace. O InnoDB sempre reserva algumas extensões para limpeza e outros propósitios internos; estas extensões reservadas não estao incluídas no espaço livre.

Quando você deletar dados de uma tabela, o InnoDB contrairá o índice de árvore-B correspondente. Ele depende do padrão de deleções se isto liberar páginas individuais ou extensões da tablespace, assim que o espaço liberado estiver disponível para outros usuários. Apagar a tabela ou deletar todos os registros dela garante a liberação do espaço para outros usuários, mas lembre-se que registros deletados só podem ser fisicamente removidos em uma operação de remoção ('purge'), depois que não houver mais necessidades de rollback em trasações ou leituras consistentes.

7.5.13.3 Desfragmentando uma Tabela

Se houver inserções ou deleções aleatórias nos índices de uma tabela, os índices podem se tornar fragmentados. Com frangmentação queremos dizer que a ordem física das páginas de índice no disco não está próxima a ordem alfabética dos registros nas páginas, ou que existe muitas páginas sem uso no bloco de 64 páginas no qual os índices são alocados.

Isto pode aumentar a varredura de índices de você usar `mysqldump` periodicamente para se fazer uma cópiad a tabela em um arquivo texto, apagar a tabela e recarregá-la a partir do arquivo texto. Outro modo de se fazer a desfragmentação é realizar uma operação alter table 'nula' `ALTER TABLE nometabela TYPE=InnoDB`. Isto faz com que o `MySQL` reconstrua a tabela.

Se as inserções a um índice são sempre crescentes e os registros só são deletados a partir do fim, então o algoritmo do gerenciamento de espaço de arquivo do InnoDB garante que a fragmentação nos índices não ocorrerão.

7.5.14 Tratando Erros

O tratamento de erro no InnoDB nem sempre é o mesmo que o especificado no padrão SQL. De acordo com o SQL-99, qualquer erro durante uma instrução SQL deve provocar o rollback da instrução. O InnoDB, algumas faz o rollback de apenas parte da instrução, ou de toda instrução. A seguinte lista especifica o tratamento de erro do InnoDB.

- Se você ficar sem espaço no tablespace você obterá do MySQL o erro `'Table is full'` e o InnoDB fará o rollback da instrução.
- Um deadlock de uma transação ou em caso de se esgotar o tempo de espera em uma trava o InnoDB fará um rollback de toda a transação.
- Um erro de chave duplicada faz um rollback da inserção deste registro em particular, mesmo em instruções como `INSERT INTO ... SELECT ...`. Caso você não especifique a opção `IGNORE` em sua instrução, provavelmente isto será diferente e o InnoDB fará rollback desta instrução SQL.
- Um erro de 'registro muito grande' faz um rollback da instrução SQL.
- Outros erros são geralmente detectado pela camada de código do MySQL e fazem o rollback da instrução correspondente.

7.5.15 Restrições em Tabelas InnoDB

- Tabelas InnoDB não suportam índices fulltext.

- No Windows o InnoDB armazena os nomes de banco de dados e tabelas internamente sempre em letras minúsculas. Para mover bancos de dados em um formato binário do Unix para o Windows ou do Windows para o Unix você deve ter todas os nomes de tabelas e banco de dados em letras minúscula.
- **Aviso**: **NÃO** converta o sistema de tabelas MySQL de MyISAM PARA InnoDB! Isto não é suportado; se você fizer isto o MySQL não reiniciará até que você restaure o sistema de tabelas antigo de um backup ou os regenere com o script `mysql_install_db`.
- `SHOW TABLE STATUS` não dá estatísticas exatas sobre tabelas InnoDB, exceto sobre o tamanho físico reservado pela tabela. O contador de linha é apenas uma estimativa rude usada na otimização SQL.
- Se você tentar criar um índice único em um prefixo de coluna você obterá um erro.

  ```
  CREATE TABLE T (A CHAR(20), B INT, UNIQUE (A(5))) TYPE = InnoDB;
  ```

  Se você criar um índice que não seja único em um prefixo de uma coluna, o InnoDB criará um índice sobre toda a coluna.
- `INSERT DELAYED` não é suportado por tabelas InnoDB.
- As operações `LOCK TABLES` do MySQL não tem conhecimento dos bloqueios de resistro do InnoDBconfigurados em instruções SQL completadas: isto significa que você pode conseguir um bloqueio de tabela mesmo se já existir transações de outros usuários que tiverem bloqueios de registros na mesma tabela. Assim suas operações sobre a tabela poder ter que esperar se eles colidirem com essas travas de outros usuários. Também pode ocorrer um deadlock. No entanto isto não tarz perigo a instegridade da transação, pois o bloqueio de registro definido pelo InnoDB sempre cuidará da integridade. Um bloqueio de tabela também previne que outras transações adquiram mais bloqueios de registros (em um modo de bloqueio conflitante) na tabela.
- Uma tabela não pode ter mais de 1000 colunas.
- `DELETE FROM TABLE` não gera a tabela novamente, mas, ao invés diato, deleta todas as linhas, uma a uma, o que não é rápido. Em versões futuras do MySQL você poderá usar `TRUNCATE` que é mais rápido.
- O tamanho de página padrão utilizado no InnoDB é 16KB. Recompilando o código pode se configurá-la com 8 KB a 64 KB. O tamanho máximo de um registro é menos da metade da página de banco de dados nas versões anteriores a 3.23.40 do InnoDB. A partir da distribuição fonte da versão 3.23.41 colunas BLOB e TEXT podem ter até 4 GB e o tamanho total do registro também devem ser menores que 4GB. O InnoDB não armazena campos cjo tamanho é menor que 128 bytes em páginas separadas. Depois do InnoDB modificar o registro armazenando campos grandes em páginas separadas, o tamanho restante da linha deve ser menor que metade da página de banco de dados. O tamanho máximo da chave é de 7000 bytes.
- Em alguns sistemas operacionais os arquivos de dados devem ser menores que 2 GB. O tamanho combinado dos arquivos de log devem ser menores que 4GB.
- O tamanho máximo do tablespace é 4 bilhões de páginas de banco de dados. Este também é o tamanho máximo da tabela. O tamanho mínimo do tabelspace é de 10 MB.
- Quando você reinicia o servidor MySQL, o InnoDB pode reutilizar um valor antigo para uma coluna `AUTO_INCREMENT`.

- Você não pode definir o primeiro valor de uma coluna `AUTO_INCREMENT` no InnoDB com `CREATE TABLE ... AUTO_INCREMENT=...` (ou `ALTER TABLE ...`). Para definir este valor insira uma linha com o valor de menos e delete esta linha.

7.5.16 Histórico de Alterações do InnoDB

7.5.16.1 MySQL/InnoDB-4.1.1, December 4, 2003

- Multiple tablespaces now available for InnoDB. You can store each InnoDB type table and its indexes into a separate '`.ibd`' file into a MySQL database directory, into the same directory where the '`.frm`' file is stored.
- The MySQL query cache now works for InnoDB tables also if `AUTOCOMMIT=0`, or the statements are enclosed inside `BEGIN ... COMMIT`.
- Reduced InnoDB memory consumption by a few megabytes if one sets the buffer pool size < 8 MB.
- You can use raw disk partitions also in Windows.

7.5.16.2 MySQL/InnoDB-4.0.16, October 22, 2003

- Fixed a bug: in contrary to what was said in the manual, in a locking read InnoDB set two record locks if a unique exact match search condition was used on a multi-column unique key. For a single column unique key it worked right.
- Fixed a bug: if one used the rename trick `#sql... -> rsql...` to recover a temporary table, InnoDB asserted in `row_mysql_lock_data_dictionary()`.
- There are several outstanding non-critical bugs reported in the MySQL bugs database. Their fixing has been delayed, because resources are allocated to the upcoming 4.1.1 release.

7.5.16.3 MySQL/InnoDB-3.23.58, September 15, 2003

- Fixed a bug: InnoDB could make the index page directory corrupt in the first B-tree page splits after '`mysqld`' startup. A symptom would be an assertion failure in '`page0page.c`', in function `page_dir_find_slot()`.
- Fixed a bug: InnoDB could in rare cases return an extraneous row if a rollback, purge, and a `SELECT` coincided.
- Fixed a possible hang over the '`btr0sea.c`' latch if `SELECT` was used inside `LOCK TABLES`.
- Fixed a bug: if a single `DELETE` statement first managed to delete some rows and then failed in a `FOREIGN KEY` error or a `Table is full` error, MySQL did not roll back the whole SQL statement as it should.

7.5.16.4 MySQL/InnoDB-4.0.15, September 10, 2003

- Fixed a bug: if you updated a row so that the 8000 byte maximum length (without `BLOB` and `TEXT`) was exceeded, InnoDB simply removed the record from the clustered

index. In a similar insert, InnoDB would leak reserved file space extents, which would only be freed at the next mysqld startup.

- Fixed a bug: if you used big `BLOB` values, and your log files were relatively small, InnoDB could in a big `BLOB` operation temporarily write over the log produced after the latest checkpoint. If InnoDB would crash at that moment, then the crash recovery would fail, because InnoDB would not be able to scan the log even up to the latest checkpoint. Starting from this version, InnoDB tries to ensure the latest checkpoint is young enough. If that is not possible, InnoDB prints a warning to the '`.err`' log of MySQL and advises you to make the log files bigger.
- Fixed a bug: setting `innodb_fast_shutdown=0` had no effect.
- Fixed a bug introduced in 4.0.13: if a `CREATE TABLE` ended in a comment, that could cause a memory overrun.
- Fixed a bug: If InnoDB printed `Operating system error number .. in a file operation` to the '`.err`' log in Windows, the error number explanation was wrong. Workaround: look at section 13.2 of http://www.innodb.com/ibman.php about Windows error numbers.
- Fixed a bug: If you created a column prefix `PRIMARY KEY` like in `t(a CHAR(200), PRIMARY KEY (a(10)))` on a fixed-length `CHAR` column, InnoDB would crash even in a simple `SELECT`. C`CHECK TABLE` would report the table as corrupt, also in the case where the created key was not `PRIMARY`.

7.5.16.5 MySQL/InnoDB-4.0.14, Junho de 2003

bullet InnoDB now supports the `SAVEPOINT` and `ROLLBACK TO SAVEPOINT` SQL statements. See http://www.innodb.com/ibman.php#Savepoints for the syntax.

bullet You can now create column prefix keys like in `CREATE TABLE t (a BLOB, INDEX (a(10)))`.

bullet You can also use `O_DIRECT` as the `innodb_flush_method` on the latest versions of Linux and FreeBSD. Beware of possible bugs in those operating systems, though.

bullet Fixed the checksum calculation of data pages. Previously most OS file system corruption went unnoticed. Note that if you downgrade from version >= 4.0.14 to an earlier version < 4.0.14 then in the first startup(s) InnoDB will print warnings:

```
InnoDB: Warning: an inconsistent page in the doublewrite buffer
InnoDB: space id 2552202359 page number 8245, 127'th page in dblwr buf.
```

but that is not dangerous and can be ignored.

bullet Modificado o algorítmo de substituição da área de buffer para que ele tente descarregar as páginas modificados se não houver páginas a serem sustituídas nos últimos 10% da lista LRU. Isto pode produzir e/s de disco se a carga de trabalho for uma mistura de leituras e escritas.

bullet O algorítmo de descarga do ponto de verificação da área de buffer agora também tenta descarregar vizinhos próximos a página no fim da lista de flush. Isto pode aumentar a velocidade de desligamento do banco de dados e pode também aumentar as escritas em disco se o arquivo de log do InnoDB for muito pequeno comparado ao tamanho da área de buffer.

bullet Na versão 4.0.13 fazemos SHOW INNODB STATUS exibir informações detalhadas sobre o último erro de UNIQUE KEY, mas armazenar esta informação podia deixar o REPLACE bem mais lento. Não exibimos nem armazenamos mais a informação.

bullet Corrigido um erro: SET FOREIGN_KEY_CHECKS=0 não era replicado apropriadamente na replicação do MySQL. A correção provavelmente não será feita na série 3.23.

bullet Corrigido um erro: o parâmetro innodb_max_dirty_pages_pct não levav em conta as páginas livres na área de buffer. Isto podia levar a descargas excessivas mesmo se houvesse muitas páginas livres na área de buffer. Solução: SET GLOBAL innodb_max_dirty_pages_pct = 100.

7.5.16.6 MySQL/InnoDB-3.23.57, June 20, 2003

bullet Changed the default value of `innodb_flush_log_at_trx_commit` from 0 to 1. If you have not specified it explicitly in your '`my.cnf`', and your application runs much slower with this new release, it is because the value 1 causes a log flush to disk at each transaction commit.

bullet Fixed a bug: InnoDB forgot to call pthread_mutex_destroy() when a table was dropped. That could cause memory leakage on FreeBSD and other non-Linux Unixes.

bullet Fixed a bug: MySQL could erroneously return 'Empty set' if InnoDB estimated an index range size to 0 records though the range was not empty; MySQL also failed to do the next-key locking in the case of an empty index range.

bullet Fixed a bug: `GROUP BY` and `DISTINCT` could treat NULL values inequal.

7.5.16.7 MySQL/InnoDB-4.0.13, 20 de Maio de 2003

bullet O `InnoDB` agora suporta `ALTER TABLE DROP FOREIGN KEY`. Você deve usar `SHOW CREATE TABLE` para ver a ID de chaves estrangeiras geradas internamente quando quiser apagar uma chave estrangeira.

bullet `SHOW INNODB STATUS` agora esxibe informações detalhadas do último erro de `FOREIGN KEY` e `UNIQUE KEY` detectados. Se você não entender porque o `InnoDB` retorna o erro 150 de um `CREATE TABLE`, você pode utilizar isto para estudar a razão.

bullet `ANALYZE TABLE` agora também funciona para tabelas do tipo `InnoDB`. Ela faz 10 inserções aleatórias para cada das árvores de índices e atualiza a estimativa da cardinalidade do índice adequadamente. Note que como isto é apenas uma estimativa, repetidas execuções de `ANALYZE TABLE` podem produzir diferentes números. O MySQL usa a estimativa de cardinalidade do índice apenas an otimização de joins. Se alguma join não é otimizada de modo apropriado, você pode tentar usar `ANALYZE TABLE`.

bullet A capacidade de commit de grupo do `InnoDB` agora também funciona quando o log binário do MySQL está habilitado. Deve haver mais de 2 threads cliente para commit de grupo estar ativo.

bullet Alterado o valor padrão de `innodb_flush_log_at_trx_commit` de 0 para 1. Se você não tiver especificado-o explicitamente em seu '`my.cnf`', e sua aplicação executar muito mais lentamente nesta nova distribuição é porque o valor 1 faz com que seja descarregado um log para disco a cada commit de transações.

bullet Adicionado uma nova variável global configurável de sistema do MySQL (`innodb_max_dirty_pages_pct`). Ela é um interio na faixa de 0 - 100. O padrão é 90. A thread principal no `InnoDB` tenta descarregar as páginas da área de buffer já que grande parte deste percetual ainda não foi descarregado em nenhum momento.

bullet Se `innodb_force_recovery=6`, não deixar o `InnoDB` fazer reparação de páginas corrompidas baseadas no buffer de dupla escrita.

bullet O `InnoDB` agora inica mais rápido porque ele não define a memória na área de buffer para zero.

bullet Corrigido um erro: a definição `FOREIGN KEY` do `InnoDB` era confudida com as palavras chaves 'foreign key' dentro dos comentários do MySQL.

bullet Corrigido um ero: se você apagasse um tablea para qual havia uma referência de chave estrangeira, e posteriormente criasse a mesma tabela com tipo de colunas não correspondentes, o `InnoDB` podia entrar em dict0load.c, na função `dict_load_table`.

bullet Corrigido um erro: `GROUP BY` e `DISTINCT` podia tratar valores `NULL` como diferentes. O MySQL também falahva ao fazer o lock da próxima chave no caso de uma faixa de índice vazia.

bullet Corrigido um erro: não faz COMMIT da transação atual quando uma tabela MyISAM é atualizada; isto também faz com que `CREATE TABLE` não faça commit de uma transação `InnoDB`, mesmo quando o log binário estiver habilitado.

bullet Corrigido um erro: não permite que `ON DELETE SET NULL` modifique a mesma tabela onde o delete foi feito; podemos permití-lo porqeu into não pode produzir loops infinitos em operações em cascata.

bullet Corrigido um erro: permitir `HANDLER PREV` e `NEXT` também depois de posicionar o cursor com uma busca única na chave primária

bullet Corrigido um erro: se `MIN()` ou `MAX()` resultasse em um deadlock ou em esgotamento do tempo de espera do lock, o MySQL não retornava um erro, mas `NULL` como o valor da função.

bullet Corrigido um erro: o `InnoDB` esquecia de chamar `pthread_mutex_destroy()` quando uma tabela era apagada. Isto podia causar perda de memória no FreeBSD e outros Unix, exceto o Linux.

7.5.16.8 MySQL/InnoDB-4.1.0, 03 de Abril de 2003

-
- O `InnoDB` agora suporta até 64 GB de memória de área de buffer em um conputador Intel de 32 bits com Windows. Isto é possível porque o `InnoDB` pode utilizar a extensão AWE de Windows para endereços de memória sobre o limite de 4 GB de um processador de 32 bits. Uma nova variável de inicialização innodb_buffer_pool_awe_mem_mb habilita o AWE e define o tamanho da área de buffer em megabytes.
- Reduz o tamanho do cabeçalho de buffer e tabela bloqueada. O `InnoDB` utiliza 2% a menos de memória.

7.5.16.9 MySQL/InnoDB-3.23.56, 17 de Março de 2003

- Corrigido um erro grave na otimização de consultas do InnoDB: consultas do tipo

SELECT ... WHERE índice_col < x and SELECT ... WHERE índice_col > x podiam provocar a varredura da tabela mesmo se a seletividade fosse muito boa.

- Corrigido um erro potencial quando MySQL chama store_lock with TL_IGNORE no meio de uma consulta.

7.5.16.10 MySQL/InnoDB-4.0.12, 18 Março de 2003

- Nas recuperações de falhas, agora o InnoDB mostra o progresso em percentual do rollback de uma transação.
- Corrigido um erro/recurso: se seu aplicativo usa mysql_use_result(), e usa >= 2 conexões para enviar consultas SQL, ele poderia entrar em deadlock na hash S-latch adaptativa em btr0sea.c. Agora o mysqld libera a S-latch se ela passar o dado de uma SELECT para o cliente.
- Corrigido um erro: o MySQL podia, erroneamente, retornar 'Empty set' se o InnoDB estimasse o tamanho da faixa do índice para 0 registro mesmo se o registro não estivesse vazio; o MySQL também falhava para fazer o lock da próxima chave no caso de uma faixa de índice vazia.

7.5.16.11 MySQL/InnoDB-4.0.11, 25 de Fevereiro de 2003

- Corrigido um erro introduzido na versão 4.0.10: SELECT ... FROM ... ORDER BY ... DESC podia entrar em loop infinito.
- Um erro proeminente: SET FOREIGN_KEY_CHECKS=0 não é replicado de forma apropriada na replicação do MySQL.

7.5.16.12 MySQL/InnoDB-4.0.10, 04 de Fevereiro de 2003

- Em INSERT INTO t1 SELECT ... FROM t2 WHERE ... anteriormente o MySQL definia um bloqueio de tabela em t2. O bloqueio agora foi removido.
- Aumentou o tamanho máximo mostardo de SHOW INNODB STATUS para 200 KB.
- Corrigido um erro grave na otimização da consulta do InnoDB: consultas do tipo SELECT ... WHERE indice_col < x and SELECT ... WHERE indice_col > x podia provocar a varredura da tabela mesmo quand a seletividade estivess muito boa.
- Corrigido um erro: a remoção ('purge') podia causar lentidão em uma tabela BLOB cuja árvore de índice de chave primária fosse de altura 1. Sintomas: os semáforos esperam devido a um tarva X definida em btr_free_externally_stored_field().
- Corrigido um erro: usar o comando HANDLER do InnoDB em um tratamento recente de um mysqld com falha em ha_innobase::change_active_index().
- Corrigido um erro: se o MySQL estimar uma consulta no meio de uma instrução SELECT, o InnoDB irá parar na trava de índice hash adaptativa em btr0sea.c.
- Corrigido um erro: O InnoDB podia relatar corrompimento e declara em page_dir_find_owner_slot() se uma busca de índice hash adaptativo coincidiu com uma remoção ou uma inserção.

- Corrigido um erro: algumas ferramentas de snapshot de sistema de arquivos no Windows 2000 podia provocar uma falha na escrita em arquivo s InnoDB com erro ERROR_LOCK_VIOLATION. Agora, em escritas síncronas, o InnoDB tenta escrever novamente até 100 vezes em intervalos de 1 segundo.
- Corrigido um erro: REPLACE INTO t1 SELECT ... não funciona se t1 tiver uma coluna com auto incremento.
- Um erro proeminente: SET FOREIGN_KEY_CHECKS=0 não é replicado de forma apropriada em replicações do MySQL.

7.5.16.13 MySQL/InnoDB-3.23.55, 24 de Janeiro de 2003

- Em INSERT INTO t1 SELECT ... FROM t2 WHERE ... anteriormente o MySQL definia um bloqueio de tabela em t2. O bloqueio agora foi removido.
- Corrigido um erro: se o tamanho total dos arquivos de log do InnoDB fosse maior que 2GB em um comoputador de 32 bits, o InnoDB escreveria o log em uma posição errada. Isto poderia fazer com que a recuperação em caso de falhas e o InnoDB Hot Backup falhassem na varredura do log.
- Corrigido um erro: restauração do cursos de índice poderia, teoricamente, falhar.
- Consrtado um erro: uma declaração em in btr0sea.c, na função btr_search_info_update_slow podia, teoriacamente, falhar em uma "disputa" de 3 threads.
- Corrigido um erro: a remoção ('purge') podia causar lentidão em uma tabela BLOB cuja árvore de índice de chave primária fosse de altura 1. Sintomas: os semáforos esperam devido a um tarva X definida em btr_free_externally_stored_field().
- Corrigido um erro: se o MySQL estimar uma consulta no meio de uma instrução SELECT, o InnoDB irá parar na trava de índice hash adaptativa em btr0sea.c.
- Corrigido um erro: O InnoDB podia relatar corrompimento e declara em page_dir_find_owner_slot() se uma busca de índice hash adaptativo coincidiu com uma remoção ou uma inserção.
- Corrigido um erro: algumas ferramentas de snapshot de sistema de arquivos no Windows 2000 podia provocar uma falha na escrita em arquivo s InnoDB com erro ERROR_LOCK_VIOLATION. Agora, em escritas síncronas, o InnoDB tenta escrever novamente até 100 vezes em intervalos de 1 segundo.
- Um erro proeminente: SET FOREIGN_KEY_CHECKS=0 não é replicado de forma apropriada em replicações do MySQL. O conserto aparecerá na versão 4.0.11 e provavelmente não será passada a versão 3.23
- Corrigido um erro na função page_cur_search_with_match em pageOcur.c do InnoDB que faz com que ele fique na mesma página indefinidamente. Este erro evidentemente só está presente em tabelas com mais de uma página.

7.5.16.14 MySQL/InnoDB-4.0.9, 14 de Janeiro de 2003

- Removida a mensagem de aviso: 'InnoDB: Out of memory in additional memory pool.'
- Corrigido um erro: se o tamanho total dos arquivos de log do InnoDB fosse maior que 2GB em um comoputador de 32 bits, o InnoDB escreveria o log em uma posição errada.

Isto poderia fazer com que a recuperação em caso de falhas e o InnoDB Hot Backup falhassem na varredura do log.

- Corrigido um erro: restauração do cursos de índice poderia, teoricamente, falhar.

7.5.16.15 MySQL/InnoDB-4.0.8, 07 de Janeiro de 2003

- Agora, o InnoDB também suporta FOREIGN KEY (...) REFERENCES ...(...) [ON UPDATE CASCADE | ON UPDATE SET NULL | ON UPDATE RESTRICT | ON UPDATE NO ACTION].
- Tabelas e índices agora reservam 4% a menos de espaço na tablespace. Tabelas existentes também reservam menos espaço. Atualizando para 4.0.8 vaocê verá mais espaço livre em "InnoDB free" em SHOW TABLE STATUS.
- Corrigido um erro: atualizar a chave primária de um registro gera uma erro de chave estrangeira em todas as chaves estrangeiras que fazem referência a chaves secundárias do registro a ser atualizado. Além disso, se uma restrição de referência de chave estrangeira só se refere a primeir coluna em um índice e houver mais colunas neste índice, atualizar a coluna adicional irá gerar um erro de chave estrangeira.
- Corrigido um erro: se um índice contém algumas colunas duas vezes e esta coluna é atualizada, a tabela se tornará corrompida. Agora o InnoDB previne a criação de tais índices.
- Corrigido um erro: removido mensagens de erros supérfluos 149 e 150 do arquivo .err quando um SELECT bloquado provoca um deadlock ou um esgota o tempo limite de espera de um bloqueio.
- Consrtado um erro: uma declaração em in btr0sea.c, na função btr_search_info_update_slow podia, teoriacamente, falhar em uma “disputa” de 3 threads.
- Corrigido um erro: não é possível trocar o nível de isolamento da tarnasação de volta para REPEATABLE READ depouis de definí-lo com outro valor.

7.5.16.16 MySQL/InnoDB-4.0.7, 26 de Dezembro de 2002

- O InnoDB na versão 4.0.7 é essencialmente o mesmo da in 4.0.6.

7.5.16.17 MySQL/InnoDB-4.0.6, 19 de Dezembro de 2002

- Uma vez que innodb_log_arch_dir não têm relevância sob o MySQL, não há necessidade de se especificá-lo no arquivo my.cnf.
- LOAD DATA INFILE em modo AUTOCOMMIT=1 não faz mais commits implicitos para cada 1MB de log binário escrito.
- Corrigido um erro introduzido na versão 4.0.4: LOCK TABLES ... READ LOCAL não deve definir bloqueio de registros ao lê-los. Isto provoca deadlocks e esgostamento do tempo limite de espera das travas do registro no mysqldump.
- Corrigido dois erros introduzidos na versão 4.0.4: em AUTO_INCREMENT, REPLACE pode fazer com que o contador pode ser deixado como 1. Um deadlock ou esgotamento do tempo limite de espera de travas podem causar o mesmo problema.

- Corrigido um erro: TRUNCATE em uma tabela temporária causa erro no InnoDB.
- Corrigido um erro introduzido na versão 4.0.5: se o log binário não estivessem ligado, INSERT INTO ... SELECT ... ou CREATE TABLE ... SELECT ... podiam fazer com que o InnoDB pendurasse em um semáforo criado em btr0sea.c, line128. Solução: ligar o log binário.
- Corrigido um erro: na replicação, executar SLAVE STOP no meio de uma transação multi-instrução podia fazer com que SLAVE START só realizasse parte da transação. Um erro parecido podia ocorrer se o slave finalizasse devido a um erro e fosse reiniciado.

7.5.16.18 MySQL/InnoDB-3.23.54, 12 de Dezembro de 2002

- Corrigido um erro: a estimativa de alcance do InnoDB exagerava em muito o tamanho de de uma pequna faixa de índice se o caminho ao ponto final da faixa na árvore de índice já era um ramo na raíz. Isto podia causar varreduras de tabela desnecessáriaem consultas SQL.
- Corrigido um erro: ORDER BY podia falhar se você não tiver criado um chave primária para um tabela, mas tiver definido diversos índices nos quais pelo menos um era um índice único (UNIQUE) com todos as suas colunas declaradas como NOT NULL.
- Corrigido um erro: um esgotamento do tempo de espera se um lock na conexão com ON DELETE CASCADE podia causar corrompimento em índices.
- Corrigido um erro: se um SELECT era feito com uma chave única a partir de um índice primário, e a busca correspondesse a um registro marcado para deleção, o InnoDB podia erroneamente retornar o PROXIMO registro.
- Corrigido um erro introduzido na versão 3.23: LOCK TABLE ... READ LOCAL não devia definir lock de registro na leitura das linhas. Isto causava deadlocks e esgotamento do tempo de espera do lock no mysqldump.
- Corrigido um erro: se um índice continha algumas colunas duas vezes, e aquela coluna está atualizada, a tabela fcava corrompida. De agora em diante o InnoDB previne a criação de tais índices.

7.5.16.19 MySQL/InnoDB-4.0.5, 18 de Novembro de 2002

- O InnoDb agora suporta os níveis READ COMMITTED e and READ UNCOMMITTED de isolmento de transação. O READ COMMITTED emula mais proximamente o Oracle e portar aplicações de Oracle para MySQL se torna mais fácil.
- A resolução de deadlock agora é seletiva: tentamos pegar como vítimas transações com menos linhas modificadas ou inseridas.
- Definições FOREIGN KEY agora está ciente da configuração lower_case_nome_tabelas no arquivo my.cnf.
- SHOW CREATE TABLE não exibe o nome do banco de dados para uma definição FOREIGN KEY se a tabela referida está no mesmo banco de dados que a tabela.
- O InnoDB faz uma verificação de consistência para verificar a maioria das páginas de índices antes de escrevê-las no arquivo de dados.

- Se você definir innodb_force_recovery > 0, o InnoDB tenta saltar para os registros e páginas com índices corrompidos fazendo SELECT * FROM tabela. Isto ajuda no dump.
- O InnoDB agora usa E/S assíncrona e sem buffer no Windows 2000 e XP; e apenas E/S sem buffer assíncrono por simulação no NT, 95/98/ME.
- Corrigido um erro: a estimativa de alcance do InnoDB exagerava em muito o tamanho de de uma pequna faixa de índice se o caminho ao ponto final da faixa na árvore de índice já era um ramo na raíz. Isto podia causar varreduras de tabela desnecessáriaem consultas SQL. A correção também será feita na versão 3.23.54.
- Corrigido um erro presente nas versões 3.23.52, 4.0.3, 4.0.4: A inicialização do InnoDB podia levar muito tempo ou até mesmo falhar em alguns computadores Windows 95/98/ME.
- Corrigido um erro: o lock AUTO-INC erá guardado para o fim da transação se ele fosse concedido depois de uma espera de lock. Isto podia causar deadlocks desnecessários.
- Corrigido um erro: se SHOW INNODB STATUS, innodb_monitor, ou innodb_lock_monitor tiver exibido centenas de transações em um relatório, e a saída ficar truncada, o InnoDB travaria, imprimindo no log de erros muitas esperas por um mutex criado em srv0srv.c, line 1621.
- Corrigido um erro: SHOW INNODB STATUS no Unix sempre relata o tamanho médio dos arquivos lidos como 0 bytes.
- Corrigido um erro potencial na versão 4.0.4: o InnoDB agora faz ORDER BY ... DESC como o MyISAM.
- Corrigido um erro: DROP TABLE podia causar falhas ou um travamento se houvesse um rollback executando concorrentemente na tabela. A correção será feita na série 3.23 se este for um problema para os usuários.
- Corrigido um erro: ORDER BY podia falhar se você não tivesse criado um chave primária para uma tabela, mas tivesse definido diversos índices nos quais pelo menos um seja um índice único (UNIQUE) com todas as suas colunas declaradas como NOT NULL.
- Corrigido um erro: um espera pelo tempo limite na conexão com ON DELETE CASCADE podia causar corrompimento nos índices.
- Corrigido um erro: se um SELECT era feito com uma chave única a partir de um índice primário e a busca correspondesse a um registro marcado para deleção, o InnoDB podia retornar o próximo registro.
- Outstanding bugs: na versão 4.0.4 dois erros foram introduzidos no AUTO_INCREMENT. REPLACE pode fazer com que o contador seja decrementado. Um deadlock ou uma espera de tempo limite de lock pode causar o mesmo problema. Eles serão corrigidos na versao 4.0.6.

7.5.16.20 MySQL/InnoDB-3.23.53, 09 de Outubro de 2002

- Usamos novamente E/S de disco sem buffer para arquivos de dados no Windows. A performance de leitura do Windows XP e Windows 2000 parecem estar muito fraca com E/S normal.

- Ajustamos a estimativa de faixa para uqe varreduras de índices na faixa tenham preferência sobre a varredura completa de índices.
- Permitir a remoção e criação de tableas mesmo se o innodb_force_recovery está configurado. Pode se usar isto para remover uma tabela que causaria uma falha no rollback ou deleção, ou se uma importação de tabelas com falhas causa um rollback na recuperação.
- Corrigido um erro presente nas versões 3.23.52, 4.0.3, 4.0.4: A inicialização do InnoDB podia demorar ou mesmo travar em alguns computadores Windows 95/98/ME.
- Corrigido um ero: a finalização rápida (que é padrão), algumas vezes ficava lenta pela união do buffer de remoção e inserção.
- Corrigido um erro: fazer um grande SELECT de uma tabela onde nenhum registro estava visível em uma leitura consistente podia causar uma espera de semáforo muito longo (> 600 segundos) em btr0cur.c line 310.
- Corrigido um erro: o lock AUTO-INC era guarda para o fim da transação se fosse concedido depois de uma espera de lock. Isto podia causar um deadlock desnecessário.
- Corrigido um erro: se você criar uma tabela temporária dentro de LOCK TABLES, e usar esta tabela temporária, causará um falha de declaração em ha_innobase.cc.
- Corrigido um erro: se SHOW INNODB STATUS, innodb_monitor, ou innodb_lock_monitor tiver exibido centenas de transações em um relatório, e a saída ficar truncada, o InnoDB travaria, imprimindo no log de erros muitas esperas por um mutex criado em srv0srv.c, line 1621.
- Corrigido um erro: SHOW INNODB STATUS no Unix sempre relata o tamanho médio dos arquivos lidos como 0 bytes.

7.5.16.21 MySQL/InnoDB-4.0.4, 02 de Outubro de 2002

- Usamos novamente E/S de disco sem buffer para arquivos de dados no Windows. A performance de leitura do Windows XP e Windows 2000 parecem estar muito fraca com E/S normal.
- Aumentado o tamanho máximo da chave de tabelas InnoDB de 500 para 1024 bytes.
- Aumentado o campo de comentário da tabela em SHOW TABLE STATUS a assim até 16000 caracteres da definição de chaves estrangeiras pode ser exibidas aqui.
- O contador de auto incremento não é mais incrementado de um inserção de uma linha falhar imediatamente.
- Permitir a remoção e criação de tableas mesmo se o innodb_force_recovery está configurado. Pode se usar isto para remover uma tabela que causaria uma falha no rollback ou deleção, ou se uma importação de tabelas com falhas causa um rollback na recuperação.
- Corrigido um erro: Usar ORDER BY primarykey DESC na versão 4.0.3 causa um falha de declaração em btr0pcur.c, line 203.
- Corrigido um ero: a finalização rápida (que é padrão), algumas vezes ficava lenta pela união do buffer de remoção e inserção.
- Corrigido um erro: fazer um grande SELECT de uma tabela onde nenhum registro estava visível em uma leitura consistente podia causar uma espera de semáforo muito longo (> 600 segundos) em btr0cur.c line 310.

- Corrigido um erro: se a cache de consultas do MySQL foi usada, ela não fica invalidada por uma modificação feita por ON DELETE CASCADE ou ...SET NULL.
- Corrigido um erro: se você criar uma tabela temporária dentro de LOCK TABLES, e usar esta tabela temporária, causará um falha de declaração em ha_innobase.cc.
- Corrigido um erro: se você definisse innodb_flush_log_at_trx_commit com 1, SHOW VARIABLES mostraria seu valor como 16 milhões.

7.5.16.22 MySQL/InnoDB-4.0.3, 28 de Agosto de 2002

- Removido um deadlock desnecessário quando a inserção precisa esperar por um lock de leitura, atualização ou deleção para liberar o lock da próxima chave.
- O comando SQL `HANDLER` do MySQL agora também funciona para os tipos de tabela `InnoDB`. O `InnoDB` faz o `HANDLER` sempre ler como leitura consistente. `HANDLER` é um caminho de acesso direto a leitura de índices individuais das tabelas. Em alguns casos `HANDLER` pode ser usado como um substituto de cursores do lado do servidor.
- Corrigido um erro na versão 4.0.2: mesmo uma única inserção podia causar um falha na versão AIX.
- Corrigido um erro: se você usar em um nome de tabela caracteres cujo código é > 127, em DROP TABLE o InnoDB podia falhar na linha 155 de pars0sym.c.
- A compilação do fonte agora fornece um versão funcional, ambas em HP-UX-11 e HP-UX-10.20. A fonte da versão 4.0.2 funciona apenas na versão 11, e a fonte do 3.23.52 apenas na 10.20.
- Corrigido um erro: se compilado em um Solaris 64-bits, o InnoDB produz um erro de bus na inicialização.

7.5.16.23 MySQL/InnoDB-3.23.52, 16 de Agosto de 2002

- O conjunto de recursos da versão 3.23 será congelada a partir desta versão. Novos recursos irão para o branch da versão 4.0, e apenas erros corrigidos serão feitos para o branch da versão 3.23.
- Muitas consultas joins no limite da CPU agora são executadas mais rápido. No Windows também muitas outras consultas no limite da CPU executar mais rápido.
- Um novo comando SQL, SHOW INNODB STATUS retorna a saída do Monitor InnoDB para o cliente. O Monitor InnoDB agora exibe informações detalhadas no último deadlock detectado.
- O InnoDB faz o otimizador de consultas SQL evitar muito mais varreduras apenas na faixa de índice e escolhe a varredura de toda a tabela. Agora isto está corrigido.
- "BEGIN" e "COMMIT" estão agora adicionados no log binário das transações A replicação do MySQL agora respeita as bordas da transação: um usuário não verá mais meia transações na replicação dos slaves.
- Um slave de replicação agora exibe na recuperação de falhas o última posição do log binário do master que ele podia recuperar.
- Uma nova configuração innodb_flush_log_at_trx_commit=2 faz o InnoDB gravar o log para uma cache de arquivo do sistema operacional a cada commit. Isto é quase

tão rápido quanto configurar innodb_flush_log_at_trx_commit=0, e configurar com 2 também tem o recurso no qual em uma falha onde o sistema operacional não teve problemas, nenhuma transação cujo commit foi realizado é perdida. Se osistema operacional falhar ou houver um queda de força,então a configurar com 2 não é mais segura que configurar com 0.

- Adicionado campos de checksum ao bloqueio de log.
- SET FOREIGN_KEY_CHECKS=0 ajuda na importação de tabelas numa ordem arbitrária que não respeita as regras de chaves estrangeiras.
- SET UNIQUE_CHECKS=0 aumenta a velocidade da importação das tabelas dentro do InnoDB se você tiver restrições de chave única em índices secundários.
- SHOW TABLE STATUS agora também lista possíveis ON DELETE CASCADE ou ON DELETE SET NULL no campo de comentário da tabela.
- Quando CHECK TABLE está executando em qualquer tipo de tabela InnoDB, ela agora verifica também o índice hash adaptativo para todas as tabelas.
- Se você definiu ON DELETE CASCADE ou SET NULL e atualizou o chave referenciada no registro pai, o InnoDB deletava ou atualizava o registro filho. Isto está alterado conforme o SQL-92: você recebe o erro 'Cannot delete parent row'.
- Melhorado o algoritmo de auto incremento: agora o primeiro inserte ou SHOW TABLE STATUS inicializa o contador de auto incremento para a tabela. Isto remove quase todos os deadlocks causados pelo SHOW TABLE STATUS.
- Alinhado alguns buffers usados na leitura e escrita dos arquivos de dados. Isto permite usar dispositivos raw sem buffer como arquivos de dados no Linux.
- Corrigido um erro: se você atualizasse a chave primária de uma tabela, podia ocorrer uma falha de declaração em page0page.ic line 515.
- Corrigido um erro: se você deleta ou atualiza um registro referenciado em uma restrição de chave estrangeira e a verificação de chave estrangeira esperapor um lock, então a verificação pode relatar um resultado errôneo. Isto também afeta a operação ON DELETE...
- Corrigido um erro: Um deadlock ou um erro de tempo esgotado na espera do lock no InnoDB causa um rollback de toda a transação, mas o MySQL ainda podia gravar as instruções SQL no log binário, embora o InnoDB faça um rollback delas. Isto podia, por exemplo, fazer a replicação do banco de dados ficar fora de sincronia.
- Corrigido um erro: se o banco de dados falha no meio de um commit, então a recuperação pode perder páginas de tablespace.
- Corrigido um erro: se você especificar um conjunto de caracteres no my.cnf, então, ao contrário do que está no manual, em uma restrição de chave estrangeira uma coluna do tipo string tinha que ter o mesmo tamanho na tabela que faz a referência e na tabela referenciada.
- Corrigido um erro: DROP TABLE ou DROP DATABASE podiam falhar se houvesse um CREATE TABLE executando simultaneamente.
- Corrigido um erro: se você configurasse a área de buffer com mais de 2GB em um computador de 32 bits, o InnoDB falharia no buf0buf.ic linha 214.

- Corrigido um erro: Em cmputadores de 64 bits,atualizando registros que contenham SQL NULL em algumas colunas faziam o undo log e o ordinary log se tornavam corrupto.
- Corrigido um erro: innodb_log_monitor causava um travamento se ele suprimisse a exibição de locks para uma página.
- Corrigido um erro: na versão HP-UX-10.20, mutexes perderiam memória e causariam condições de corrida e falhariam em alguma parte do código do InnoDB.
- Corrigido um erro: se você rodou em modo AUTOCOMMIT, executou um SELECT, e imeditamente depois um RENAME TABLE, então RENAME falharia e o MySQL reclamaria com o erro 192.
- Corrigido um erro: se compilado no Solaris 64 bits, o InnoDB produiria um erro de bus na inicialização.

7.5.16.24 MySQL/InnoDB-4.0.2, 10 de Julho de 2002

- InnoDB is essentially the same as InnoDB-3.23.51.
- If no innodb_data_file_path is specified, InnoDB at the database creation now creates a 10 MB auto-extending data file ibdata1 to the datadir of MySQL. In 4.0.1 the file was 64 MB and not auto-extending.

7.5.16.25 MySQL/InnoDB-3.23.51, 12 de Junho de 2002

- Corrigido um erro: uma join podia resultar em um segmentation faut ao copiar de uma coluna BLOB para TEXT se alguma das colunas BLOB ou TEXT na tabela continham um valor NULL do SQL.
- Corrigido um erro: se você adicionasse restrições de chaves estrangeiras auto referenciais com ON DELETE CASCADE a tabelas e uma deleção de registro fazia o InnoDB tentar deletar o mesmo registro duas vezes devido a deleção em cascata e então você obtinha um falha de declaração.
- Corrigido um erro: se você usar o 'lock de usuário' do MySQL e fechasse uma conexão, então o InnoDB podia falhar em ha_innobase.cc, line 302.

7.5.16.26 MySQL/InnoDB-3.23.50, 23 de Abril de 2002

- O InnoDB agora suporta uma auto extensão do último arquivo de dados. Você não precisa prealocar todos os arquivos de dados na inicialização do banco de dados.
- Faz diversas alterações para facilitar o uso da ferramenta Hot Backup do InnoDB. Esta é uma ferramenta separada paga que você pode usar para tirar backus online do seu banco de dados se desligar o servidor ou configurar qualquer lock.
- Se você quiser executar a ferramenta Hot Backup do InnoDB em um arquivo de dados auto extendido você terá que atualizá-lo para a versão ibbackup-0.35.
- A fase de varredura do log na recuperação de falhas agora executará muito mais rápido.
- A partir desta versão do servidor, a ferramenta de hot backup trunca os fins dos arquivos de dados do backup do InnoDB inutilizados.

- Para permitir que a ferramenta de hot backp funcione, no Windows não usaremos mais E/S sem buffer ou E/S assíncrona nativa; usaremos a mesma assincronia simulada como no Unix.
- Agora você pode definir as cláusulas ON DELETE CASCADE ou ON DELETE SET NULL em caves estrangeiras.
- Restrições de chaves estrangeiras agora sobrevivem a ALTER TABLE e e CREATE INDEX.
- Suprimimos a verificação de FOREIGN KEY se qualquer um dos valores de coluna na chave estrangeira ou chave referenciada a ser verificada é SQL NULL. Isto écompatível com Oracle, por exemplo.
- SHOW CREATE TABLE agora também lista todas as restrições de chaves estrangeiras. O mysqdump também não esquece mais sobre sobre chaves estrangeiras na definiçãode tabelas.
- Agora você pode adicionar uma nova restrição de chave estrangeira com ALTER TABLE ... ADD CONSTRAINT FOREIGN KEY (...) REFERENCES ... (...).
- As definições de FOREIGN KEY agora permitem nomes de tabela e colunas entre aspas invertidas.
- O comando MySQL SET TRANSACTION ISOLATION LEVEL ... agora tem o seguinte efeito em tabelas InnoDB: se uma transação é definida como SERIALIZABLE então o InnoDB conceitualmente adiciona LOCK IN SHARE MODE para todas as leituras consistentes. Se uma transação é definida com qualquer outro nível de isolação, então o InnoDB obedece sua estratégia de lock padrão que é REPEATABLE READ.
- SHOW TABLE STATUS não configuram mais um x-lock no fim de um índice auto incremento se um contador auto incremento já tiver sido inicializado. Isto remove quase todos os casos de deadlock causados por SHOW TABLE STATUS.
- Corrigido em erro: em uma instrução CREATE TABLE statement a string 'foreign' seguida por caracter que não seja de espaço confuder o analizador do FOREIGN KEY e faz a criação de tabelas falhar com número de erro 150.

7.5.16.27 MySQL/InnoDB-3.23.49, 17 de Fevereiro de 2002

- Corrigido um erro: se você chamasse DROP DATABASE para um banco de dados no qual haviam consultas executando simultaneamente, o MySQL podia falhar ou travar. A falha foi corrigida, mas uma correção completa terá que esperar por alguas mudanças na camada de código do MySQL.
- Corrigido um erro: no Windows deve se colocar o nome do banco de dados em minúsculo para DROP DATABASE funcionar. Corrigido na versão 3.23.49: o caso não é mais problema no Windows. No Unix o nome de banco de dadospermanece caso sensitivo.
- Corrigido um erro: se se definisse um conjunto de caracteres diferente de latin1 como o conjunto de caracteres padrão, então a definição das restrições de chaves estrangeiras podiam falhar em uma declaração em dict0crea.c, relatando um erro interno 17.

7.5.16.28 MySQL/InnoDB-3.23.48, 09 de Fevereiro de 2002

- Ajustado o otimizador SQL para favorecer busca de índices sobre a varredura de tabelas

com mais frequencia.

- Corrigido um problema de performance quando diversas consultas SELECT grandes estão executando concorrentemente em um computados Linux multiprocessador. Grandes consultas SELECT no limite da CPU tambe serão executadas mais rapído em todas as plataformas de uma maneira geral.
- Se olog binário do MySQL é usado, o InnoD agora exibe, após a recuperação de falhas, o nome do último arquivo de log binário do MySQL e a posição neste arquivo (=byte offset) que o InnoDB pode recuperar. Isto é útil, por exemplo, quando sincronizar um banco de dados master e um slave na replicação novamente.
- Adicionado uma mensagem de erro melhor para ajudar nos problemas de instalação.
- Pode-se agora recuperar também tabelas temporárias do MySQL que se tronaram órfão dentro do tablespace do InnoDB.
- O InnoDB agora previne que uma declaração FOREIGN KEY onde o sinal não é o mesmo nas colunas inteiras de referência e referenciada.
- Corrigido um erro: chamar SHOW CREATE TABLE ou SHOW TABLE STATUS poderia causar corrompimento de memória e fazer o mysqld falhar. O mysqldump, especialmente, corria este risco, já que ele chamava SHOW CREATE TABLE com frequencia.
- Corrigido um erro: se no Unix você fazia um ALTER TABLE em uma tabela e, simultaneamente, executava consultas nela, o mysqld podia falhar em uma declaração no row0row.c, linha 474.
- Corrigido um erro: se inserir diversas tabelas contendo uma coluna auto incremento estava envolvida dentro do LOCK TABLES, o InnoDB falhava em lock0lock.c.
- A versão 3.23.47 permitia diversos NULLS em um índice secundário UNIQUE. Mas CHECK TABLE não era relaxed: ele rporta atabela como corrompida. CHECK TABLE não reclama mais nesta situação.
- Corrigido um erro: no Sparc e outros processadores high-endian, SHOW VARIABLES exibia innodb_flush_log_at_trx_commit e outros parâmetros de inicialização booleanos sempre como OFF mesmo se eles estivessem habiliados.
- Corrigido um erro: se você executava mysqld-max-nt como um serviço no Windows NT/2000, a finalização do serviço nãoesperava o suficiente que o desligamento do InnoDb finalizasse.

7.5.16.29 MySQL/InnoDB-3.23.47, 28 de Dezembro de 2001

- A recuperação agora é mais rápida, especialmente em um sistema de carga leve, pois a verificação do background tem sido feita com mais frequencia.
- O InnoDB permite agora diversos valores de chaves parecidas em um índice secundário UNIQUE se aqueles valores contêm NULLs do SQL. Assim a convenção agora é a mesma das tabelas MyISAM.
- O InnoDB traz uma melhor estimativa de contagem de linhas de uma tabela contendo BLOBs.
- Em uma restrição FOREIGN KEY, o InnoDB agora é caso insensitivo para nomes de colunas e no Windows para nome de tabelas também.

- O InnoDB permite uma coluna FOREIGN KEY do tipo CHAR se referir a uma coluna do tipo VARCHAR e vice versa. O MySQL silenciosamente troca os tipos de algumas colunas entre CHAR e VARCHAR e estas alterações silenciosas não seguem declarações de FOREIGN KEY mais.
- A recuperação era mais suceptível ao corrompimento de arquivos de log.
- Cálculo de estatísticas desnecessárias forma removidas das consultas que geravam um tabea temporária. Algumas consultas ORDER BY e DISTINCT executarão muito mais rápido agora.
- O MySQL agora sabe que a varredura de uma tabela InnoDB é feita através de uma chave primária. Isto economizará uma ordenação em algumas consultas ORDER BY.
- O tamanho máximo da chave de tabelas InnoDB está restrita novamente a 500 bytes. O interpretador do MySQL não pode tratar chaves longas.
- O valor padrão de innodb_lock_wait_timeout foi alterado de infinito para 50 segundos, e o valor padrão de innodb_file_io_threads de 9 para 4.

7.5.16.30 MySQL/InnoDB-4.0.1, 23 de Dezembro de 2001

- O InnoDB é o mesmo da versão 3.23.47.
- Na versão 4.0.0 o interpretador do MySQL não conhece a sintaxe de LOCK IN SHARE MODE. Isto foi corrigido.
- Na versão 4.0.0 deleções multi-tabelas não funcionavam para tabelas transacinais, Isto foi corrigido.

7.5.16.31 MySQL/InnoDB-3.23.46, 30 de Novembro de 2001

- É o mesmo da versão 3.23.45.

7.5.16.32 MySQL/InnoDB-3.23.45, 23 de Novembro de 2001

- Esta é uma distribuição para correção de erros.
- Nas versões 3.23.42-.44, ao criar uma tabela no Windows você tinha que usar letras minúsculas nos nomes de bancos de dados para poder acessara tabela. Corrigido na versão 3.23.45.
- O InnoDB agora descarrega stdout e stderr a cada 10 segundos; se eles estiverem redirecionados para arquivos, o conteúdo do arquivo pode ser melhor vizualizado com um editor.
- Corrigida um falha em .44, in trx0trx.c, linha 178 quando você removia uma tabela cujo o arquivo .frm não existia dentro do InnoDB.
- Corrigido um erro no buffer de inserção. A árvore do buffer de inserção podia entrar em um estado de inconsistência, causando uma falha, e também falhara recuperação, Este erro podia aparecer, especialmente, em importação de grandes tabelas ou alterações.
- Corrigido um erro na recuperação: o InnoDB podia entrar em loop infinito constantemente exibindo uma mensagem de aviso de que ele não podia encontrar blocos livres na área de buffer.

- Corrigido um erro: quando você criava uma tabela temporária de um tipo InnoDB e então usava ALTER TABLE para ela,o servidor MySQL podia falhar.
- Previnia a criação das tabeas do sistema do MySQL, 'mysql.user', 'mysql.host', ou 'mysql.db', no tipo InnoDB.
- Corrigido um erro que podia causar um falha de declaração na versão 3.23.44 em srv0srv.c, linha 1728.

7.5.16.33 MySQL/InnoDB-3.23.44, 02 de Novembro de 2001

- Você pode definir restrições de chaves estrangeiras em tabelas InnoDB. Um exemplo: FOREIGN KEY (col1) REFERENCES table2(col2).
- Você pode criar arquivos de dados > 4 GB naqueles sistems de arquivos que permitem isto.
- Melhorado os monitores do InnoDB, incluindo um novo innodb_table_monitor que permite que você mostre o conteúdo do dicionário de dados interno do InnoDB.
- DROP DATABASE funcionará também com tabelas InnoDB.
- Caracteres de acento no conjunto de caracteres padrão latin1 serão ordenados de acordo com com a ordenação do MySQL. NOTA: se você está usando o latin1 e inseriu caracteres cujo código é > 127 em uma coluna CHAR indexada, você deve executar CHECK TABLE em sua tabela quando atualizar para a versão 3.23.43, e remover e reimportar a tabela se CHECK TABLE relatar um erro. reports an error!
- O InnoDB calculará melhor a estmativa da cardinlidade da tabela.
- Alteração na resolução do deadlock: na versão 3.23.43 um deadlock fazia rolls back apenas nas instruções SQL, 3.23.44 faz o rollback de toda a transação.
- Deadlock, esgotamento do tempo de espera do lock e violação das restrições de chave estrangeiras (sem registro pais e registros filhos existentes) agora retorna códigos de erro nativos do MySQL 1213, 1205, 1216, 1217, respectivamente.
- Um novo parâmetro do my.cnf (innodb_thread_concurrency) ajuda no ajuste de performance em ambientes de alta concorrência.
- Uma nova opção do my.cnf (innodb_force_recovery) lhe ajuda no dump de tabelas de um banco de dados corrompidos.
- Uma nova opção do my.cnf (innodb_fast_shutdown) aumentará a velocidade do desligamento. Normalmente o InnoDB faz uma união total dos buffers de inserção e remoção na finalização.
- Aumentado o tamanho máximo da chave para 7000 bytes de um tamanho anterior de 500 bytes.
- Corrigido um erro na replicação de colunas auto-incremento com inserção de multiplas linhas.
- Corrigido um erro quando o caso das letras alteram em uma atualização de uma coluna de índice secundário.
- Corrigido uma trava quando havia > 24 arquivos de dados.
- Corrigido uma falha quando MAX(col) é selecionado de uma tabela vazia, e col é uma coluna diferente da primeira em um índice multi-colunas.
- Corrigido um erro na remoção que podia causar falhas.

7.5.16.34 MySQL/InnoDB-3.23.43, 04 de Outubro de 2001

- Ele é essencialmente o mesmo que o InnoDB-3.23.42.

7.5.16.35 MySQL/InnoDB-3.23.42, 09 de Setembro de 2001

- Corrigido um erro que corrompia a tabela se a chave primária de um registro com mais de 8000-byte fosse atualizado.
- Existem 3 tipos de InnoDB Monitors: innodb_monitor, innodb_lock_monitor, and innodb_tablespace_monitor. Agora o innodb_monitor também mostra a taxa de acerto da área de buffer e o total de registros inseridos, atualizados, deletados e lidos.
- Corrigido um erro em RENAME TABLE.
- Arruamdo um erro em replicação com uma coluna auto-incremento.

7.5.16.36 MySQL/InnoDB-3.23.41, 13 de Agosto de 2001

- Suporte para < 4 GB de registros. O limite anterior era de 8000 bytes.
- Usa o método de descarga do arquivo de dupla escrita.
- Partições de disco raw suportadas como arquivos de dados.
- InnoDB Monitor.
- Diversos erros corrigidos um erro em `ORDER BY` ('Sort aborted') arrumado.

7.5.16.37 MySQL/InnoDB-3.23.40, 16 de Julho de 2001

- Apenas alguns erros raros foram concertados

7.5.16.38 MySQL/InnoDB-3.23.39, 13 de Junho de 2001

- Agora `CHECK TABLE` funciona em tabelas `InnoDB`.
- Um novo parâmetro `innodb_unix_file_flush_method` em '`my.cnf`' é introduzido. Ele pode ser usado para sintonizar o desempenho da escrita em disco.
- Uma coluna auto-increment agora obtem novos valores antes do mecanimo de transação. Isto economiza tempo de CPU e elimina deadlocks em transações em atribuições de novos valores.
- Diversos erros arrumados, o mais importante é o erro de rollback na 3.23.38.

7.5.16.39 MySQL/InnoDB-3.23.38, 12 de Maio de 2001

- A nova sintaxe `SELECT ... LOCK IN SHARE MODE` é introduzida.
- O `InnoDB` agora chama `fsync` depois de cada escrita em disco e clacula um checksum para todas as páginas do banco de dados que ele escreve ou lê, revelando defeitos e disco.
- Diversos erros arrumados.

7.5.17 Informações de Contato do InnoDB

Informções para contato do Innobase Oy, produtor do mecanismo InnoDB. Web site: http://www.innodb.com/. E-mail: sales@innodb.com

```
phone: 358-9-6969 3250 (office) 358-40-5617367 (mobile)
Innobase Oy Inc.
World Trade Center Helsinki
Aleksanterinkatu 17
P.O.Box 800
00101 Helsinki
Finland
```

7.6 Tabelas BDB ou BerkeleyDB

7.6.1 Visão Geral de Tabelas BDB

BerkeleyDB, disponível em http://www.sleepycat.com/ tem provido o MySQL com um mecanismo de armazenamento transacional. O suporte para este mecanismo de armazenamento está incluído na distribuição fonte do MySQL a partir da versão 3.23.34 e está ativo no binário do MySQL-Max. Este mecanismo de armazenamento é chamado normalmente de BDB.

Tabelas BDB podem ter maior chance de sobrevivência a falhas e também são capazes de realizar operações COMMIT e ROLLBACK em transações. A distribuição fonte do MySQL vem com uma distribuição BDB que possui alguns pequenos patchs para faze-lo funcionar mais suavemente com o MySQL. Você não pode usar uma versão BDB sem estes patchs com o MySQL.

Na MySQL AB, nós estamos trabalhando em cooperação com a Sleepycat para manter a alta qualidade da interface do MySQL/BDB.

Quando trouxemos o suporte a tabelas BDB, nos comprometemos a ajudar os nosso usuários a localizar o problema e criar um caso de teste reproduzível para qualquer problema envolvendo tabelas BDB. Tais casos de teste serão enviados a Sleepycat que nos ajudará a encontrar e arrumar o problema. Como esta é uma operação de dois estágios, qualquer problema com tabelas BDB podem levar um tempo um pouco maior para ser resolvido do que em outros mecanismos de armazenamento. De qualquer forma, como o código do BerkeleyDB tem sido usado em autras aplicações além do MySQL, nós não vemos nenhum grande problema com isto. Veja Seção 1.4.1 [Suporte], Página 17.

7.6.2 Instalando BDB

Se você tiver feito o download de uma versão binária do MySQL que inclui suporte a BerkeleyDB, simplesmente siga as instruções de instalação de uma versão binária do MySQL. Veja Seção 2.2.9 [Instalando o binário], Página 91. Veja Seção 4.8.5 [mysqld-max], Página 343.

Para compilar o MySQL com suporte a BerkeleyDB, faça o download do MySQL versão 3.23.34 ou mais novo e configure MySQL com a opção --with-berkeley-db. Veja Seção 2.3 [Instalando o fonte], Página 94.

```
cd /path/to/source/of/mysql-3.23.34
./configure --with-berkeley-db
```

Por favor, de uma olhada no manual fornecido com a distribuição `BDB` para informações mais atualizadas.

Mesmo sendo o BerkeleyDB muito testado e confiável, a interface com o MySQL ainda é considerada com qualidade gamma. Nós estamos ativamente melhorando e otimizando para torná-la estável o mais breve possível.

7.6.3 Opções de Inicialização do `BDB`

Se você estiver executando com `AUTOCOMMIT=0` então as suas alterações em tabelas `BDB` não serão atualizadas até que você execute um `COMMIT`. No lugar de commit você pode executar um `ROLLBACK` para ignorar as suas alterações. Veja Seção 6.7.1 [COMMIT], Página 615.

Se você estiver execuando `AUTOCOMMIT=1` (padrão), será feito um commit das sua alterações imediatamente. Você pode iniciar uma transação estendida com o comando SQL `BEGIN WORK`, depois do qual não será feito commit de suas alterações ae que você execute `COMMIT` (ou faça `ROLLBACK` das alterações.)

As seguintes opções do `mysqld` podem ser usadas pa alterar o comportamento de tabelas `BDB`:

Opção	Descrição
`--bdb-home=directory`	Diretório base das tabelas `BDB`. Ele deve ser o mesmo diretório usado para `--datadir`.
`--bdb-lock-detect=#`	Detecção de travas de Berkeley. Pode ser (`DEFAULT`, `OLDEST`, `RANDOM`, ou `YOUNGEST`).
`--bdb-logdir=directory`	Diretório de arquivos log de Berkeley DB.
`--bdb-no-sync`	Não sincroniza logs descarregados.
`--bdb-no-recover`	Não inicia Berkeley DB no modo de recuperação.
`--bdb-shared-data`	Inicia Berkeley DB no modo de multi-processos (Não usa `DB_PRIVATE` ao inicializar Berkeley DB)
`--bdb-tmpdir=directory`	Diretorio de arquivos temporários do Berkeley DB.
`--skip-bdb`	Disabilita o uso de tabelas `BDB`.
`-O bdb_max_lock=1000`	Define o número máximo de travas possíveis. Veja Seção 4.6.8.4 [bdb_max_lock], Página 309.

Se você utiliza `--skip-bdb`, MySQL não irá inicializar o biblioteca Berkeley DB e isto irá economizar muita memória. É claro que você não pode utilizar tabelas `BDB` se você estiver usando esta opção. Se você tentar criar uma tabela `BDB`, o MySQL criará uma tabela `MyISAM`.

Normalmente você deve iniciar `mysqld` sem `--bdb-no-recover` se você pretende usar tabelas `BDB`. Isto pode, no entanto, lhe trazer problemas quando você tentar iniciar o `mysqld` e os arquivos de log do `BDB` estiverem corrompidos. Veja Seção 2.4.2 [Iniciando o servidor], Página 116.

Com `bdb_max_lock` você pode especificar o número mácimo de travas (10000 por padrão) que você pode tar ativas em uma tabela `BDB`. Você deve aumentá-lo se você obter um erro do tipo `bdb: Lock table is out of available locks` ou `Got error 12 from ...` quando você fizer transações longas ou quando `mysqld` tiver que examinar muitas linhas para calcular a consulta.

Você também pode desejar alterar `binlog_cache_size` e `max_binlog_cache_size` se você estiver usando transações multi-linhas. Veja Seção 6.7.1 [COMMIT], Página 615.

7.6.4 Características de Tabelas `BDB`:

- Para estar apto a fazer roolback da transação, o mecanismo de armazenamento `BDB` mantém arquivos de log. Para obter o máximo de desempenho você deve colocar estes arquivos em outro disco diferente do usado por seus bancos de dados usando a opção `--bdb-logdir`.
- O MySQL realiza um ponto de verificação a cada vez que um novo arquivo de log do `BDB` é iniciado e remove qualquer arquivo de log que não for necessário para a transação atual. Pode se executar `FLUSH LOGS` a qualquer momento para faze um ponto de verificação de tabelas Berkeley DB.

  Para recuperação de desastres, deve-se usar backups de tabelas mais log binário do MySQL. Veja Seção 4.5.1 [Backup], Página 275.

  Aviso: Se você delatar arquivos de log antigos que estão em uso, o `BDB` não estará apto a fazer a recuperação e você pode perder dados se algo der errado.
- O MySQL precisa de uma `PRIMARY KEY` em cada tabela `BDB` para poder fazer referência a linha lida anteriormente. Se você não criar um o MySQL criará uma chave primária oculta para você. A chave oculta tem um tamanho de 5 bytes e é incrementada a cada tentaiva de inserção.
- Se todas as colunas que você acessa em uma tabela `BDB` são parte do mesmo índice ou parte de uma chave primária, então o MySQL pode executar a consulta ser ter que acessar a linha atual. Em uma tabela `MyISAM` o descrito acima é guardado apenas se as colunas são parte do mesmo índice.
- A `PRIMARY KEY` será mais rápida que qualquer outra chave, já que a `PRIMARY KEY` é armazenada junto com o registro do dado. Como as outras chaves são armazenads como os dados da chave `+` a `PRIMARY KEY`, é importante manter a `PRIMARY KEY` o menor possível para economizar disco e conseguir maior velocidade.
- `LOCK TABLES` funciona em tabelas `BDB` como nas outras tabelas. Se você não utilizar `LOCK TABLE`, MySQL comandará um bloqueio interno de múltipla-escrita nas tabelas para assegurar que a tabela será bloqueada apropriadamente se outra thread executar um bloqueio de tabela.
- Bloqueios internos em tabelas `BDB` é feito por página.
- `SELECT COUNT(*) FROM nome_tabela` é lento pois tabelas `BDB` não mantém um contador do número de linha na tabela.
- A varredura sequencial é mais lenta que com tabelas `MyISAM` já que os dados em tabelas `BDB` são armazenados em árvores-B e não em um arquivo de dados separado.

- A aplicação sempre deve estar preparada para tratar casos onde qualquer alteração de uma tabela `BDB` pode fazer um rollback automático e quqlquer leitura pode falhar com um erro de deadlock.
- As chaves não são compactadas por prefixo ou por sufixo como em tabelas `MyISAM`. Em outras palavras, a informação da chave gastará um pouco mais de espaço em tabelas `BDB` quando comparadas a tabelas `MyISAM`.
- Existem buracos frequentemente em tabelas `BDB` para permitir que você insira novas linhas no meio da árvore de chaves. Isto torna tabelas `BDB` um pouco maiores que tabelas `MyISAM`.
- O otimizador precisa conhecer aproximadamente o número de linhas na tabela. O MySQL resolve isto contando inserções e mantendo isto em um segmento separado em cada tabela `BDB`. Se você não executar várias instruções `DELETE` ou `ROLLBACK`, este número deverá estar suficientemente próximo do exato para o otimizador do MySQL, mas como o MySQL só armazerna o número ao finalizar, ele pode estar incorreto se o MySQL finalizar inesperadamente. Isto não deve ser fatail mesmo se este número não for 100% correto. POde se atualizar o número de linhas executando `ANALYZE TABLE` ou `OPTIMIZE TABLE`. Veja Seção 4.6.2 [ANALYZE TABLE], Página 298 . Veja Seção 4.6.1 [OPTIMIZE TABLE], Página 298.
- Se você ficar com o seu disco cheio com uma tabela `BDB`, você obterá um erro (provavelmente erro 28) e deve ser feito um rollback da transação. Isto está em contraste com as tabelas `MyISAM` e `ISAM` onde o `mysqld` irá esperar por espaço suficiente em disco pra continuar.

7.6.5 Itens a serem corrigidos no `BDB` num futuro próximo:

- É muito lento abrir muitas tabelas `BDB` ao mesmo tempo. Se você for utilizar tabelas `BDB`, você não deve ter um cache de tabela muito grande (`>` 256) e você deve usar `--no-auto-rehash` com o cliente `mysql`. Nós planejamos arrumar isto parcialmente na versãp 4.0.
- `SHOW TABLE STATUS` ainda mão fornece muitas informações para tabelas `BDB`
- Otimizar o desempenho.
- Fazer com que não seja utilizado bloqueio de páginas quando varrermos a tabela.

7.6.6 Sistemas operacionais suportados pelo `BDB`

Atualmente sabemos que o mecanismo de armazenamento `BDB` funciona com os seguintes sistemas operacionais:

- Linux 2.x Intel
- Sun Solaris (sparc e x86)
- FreeBSD 4.x/5.x (x86, sparc64)
- IBM AIX 4.3.x
- SCO OpenServer
- SCO UnixWare 7.0.1

Ele não funciona com os seguintes sistemas operacionais.

- Linux 2.x Alpha
- Linux 2.x AMD64
- Linux 2.x IA64
- Linux 2.x s390
- Max OS X

Nota: A lista acima não está completa; atualizaremos ela assim que recebermos mais informações.

Se você construir o MySQL como suporte a tabelas `BDB` e obter o seguinte erro no arquivo de log quando você iniciar o `mysqld`:

```
bdb: architecture lacks fast mutexes: applications cannot be threaded
Can't init dtabases
```

Isto significa que as tabelas `BDB` não são suportadas por sua arquitetura. Neste caso você deve reconstruir o MySQL sem o suporte a tabelas `BDB`.

7.6.7 Restrições em Tabelas `BDB`

Aqui segue as restrições que você tem quando utiliza tabelas `BDB`:

- Tabelas `BDB` armazenam no arquivo '`.db`' o caminho para o arquivo no qual ela foi crada. (Isto foi feito para tornar possível detectar travas em um ambiente multi-usuário que suporte links simbólicos)

  O efeito disto é que tabelas `BDB` não podem ser movidas entre diretórios!
- Ao tirar backups de tabelas `BDB`, você pode utilizar `mysqldump` ou tirar backup de todos os arquivos `nome_tabela.db` e os arquivos de log do `BDB`. Os arquivos de log do `BDB` são os arquivos no diretório de dados base chamado `log.XXXXXXXXXX` (dez digitos); O mecanismo de armazenamento `BDB` guarda transações não terminadas em arquivos de log e exige que estes arquivos sejam apresentados quando o `mysqld` iniciar.

7.6.8 Erros Que Podem Ocorrer Usando Tabelas `BDB`

- Se você obter o seguinte erro no log `hostname.err` ao iniciar o `mysqld`:

  ```
  bdb:  Ignoring log file: .../log.XXXXXXXXXX: unsupported log version #
  ```

  significa que a nova versão `BDB` não suporta o formato do arquivo de log antigo. Neste caso você tem que deletar todos os logs `BDB` do seu diretório de banco de dados (o arquivo com nomes no formato `log.XXXXXXXXXX`) e reiniciar o `mysqld`. Também recomentdamos que você faça um `mysqldump --opt` de sua tabela `BDB` antiga, delete as tabelas antigas e restaure o dump.
- Se você não estiver executando em modo auto-commit e deltar uma tabela que é referenciada em outra transação, você pode obter a seguinte mensagem de erro em seu log de erro do MySQL:

  ```
  001119 23:43:56  bdb:  Missing log fileid entry
  001119 23:43:56  bdb:  txn_abort: Log undo failed for LSN:
                         1 3644744: Invalid
  ```

Isto não é fatal mas não recomendamos deletar tabelas se não estiver no modo auto-commit, até que este problema seja resolvido (a solução não é trivial).

8 Introdução ao MaxDB

MaxDB é um banco de dados empresarial. O MaxDB é o novo nome de um sistema de gerenciamento de banco de dados formalmente chamado SAP DB.

8.1 Historia do MaxDB

A história do SAP DB vem do ínicio dos anos 80, quando ele foi desenvolvido como um produto comercial (Adabas). O banco de dados mudou de nome diversas vezes desde então. Quando a SAP AG, uma companhia Alemã tomou conta do desenvolvimento deste sistema de banco de dados, ele foi chamado de SAP DB.

A SAP desenvolve sistemas de banco de dados para servir como um sistema de armazenamento para todas as aplicações pesadas SAP, chamadas R/3. O SAP DB foi criado para fornecer uma alternativa para sistemas de banco de dados como o Oracle, Microsoft SQL Server, ou DB2 da IBM. Em Outubro de 2000, A SAP AG liberou o SAP DB soba licença GNU GPL (veja Apêndice H [GPL license], Página 1096), fazendo dele um programa open source. Em Outubro de 2003, mais de 2000 clientes da SAP AG estavam usando SAP DB como o seu principal sistema de banco de dados, e mais de outros 2000 clientes o estavam usando como um sistema de banco de dados a parte além do banco de dados principal, como parte de uma solução APO/LivaCache.

Em Maio de 2003, uma parceria foi formada entre a MySQL AB e a SAP AG. Esta parceria permite à MySQL AB desenvolver no SQP DB, renomeá-lo e a vender licenças comerciais do SAP DB para clientes que não queiram ser limitados pelas restrições impostas a eles quando usam o sistema de banco de dados sob uma licença GNU GPL (veja Apêndice H [GPL license], Página 1096). Em Agosto 2003, o SAP DB foi renomeado para MaxDB pela MySQL AB.

8.2 Licenciamento e Suporte

O MaxDB pode ser usado sob as mesmas licenças disponíveis para os outros produtos distribuídos pela MySQL AB (veja Seção 1.4.3 [MySQL licenses], Página 18). Assim, o MaxDB estará disponível sob a GNU General Public License (veja Apêndice H [GPL license], Página 1096), e uma licença comercial (veja Seção 1.4 [Licensing and Support], Página 17).

A MySQL irá oferecer suporte para MaxDB para clientes não-SAP.

A primeira versão renovada será o MaxDB 7.5.00 que será liberada no fim de 2003.

8.3 Conceitos Básicos do MaxDB

O MaxDB opera como um produto cliente/servidor. Ele foi desenvolvido para cobrir a demanda de instalações que processam um alto volume de transações on line. Tanto a expansão quanto o backup online do banco de dados são suportados. O Microsoft Clustered Server é suportado diretamente para implementações multi-servidor; outras soluções para falhas devem ser feitas manualmente. A ferramenta de gerenciamento de banco de dados são fornecidos tanto na implementação Windows quanto na baseada em browser.

8.4 Diferenças de Recursos entre o MaxDB e o MySQL

A lista a seguir fornece um pequeno resumo das principais diferenças entre o MaxDB e o MySQL; ela não esta completa.

- MaxDB funciona como um sistema cliente/servidor. O MySQL pode fincionar como um sistema cliente/servidor ou como um sistema embutido.
- O MaxDB não pode ser executado em todas as plataformas suportadas pelo MySQL. Por exemplo, o MaxDB não funciona no OS/2 da IBM.
- O MaxDB usa um protocolo de rede proprietário para comunicação cliente servidor, enquanto o MySQL usa o TCP/IP (com ou sem criptografia SSL), sockets (sob sistemas do tipo Unix) ou named pipes (sob sistemas da familia Windows-NT).
- O MaxDB suporta stored procedures. Para o MySQL, stored procedures não estão programadas para implementação até a versão 5.0. O MaxDB também suporta programação de triggers por meio de extensão SQL, que está previsto para o MySQL 5.1. O MaxDB contém um depurador para linguagens com stored procedures, pode fazer cascade de triggers aninhados e suporta vários triggers por ação e linha.
- O MaxDB é distribuído com interface de usuários em modo texto, gráfico ou baseado web. O MySQL é distribuído apenas com interfaces de usuários em modo text; uma interface gráfica do usuário (MySQL Control Center) é distribuída separadamente da distribução principal. Interfaces com o usuários baseada em Web para o MySQL são ofereceidas por terceiros.
- O MaxDB suporta um número de interfaces de programação também suportadas pelo MySQL. No entanto, o MaxDB não suporta RDO, ADO, ou .NET, os quais são suportadas pelo MySQL. O MaxDB suporta SQL embarcado apenas com C/C++.
- O MaxDB contém recursos administrativos que o MySQL não tem: Agendamento de tarefas por hora, evento, e alerta , e permite enviar mensagens para um administrador de banco de dados nos avisos.

8.5 Interoperability Features between MaxDB and MySQL

Os seguintes recursos serão incluídos nas versão do MaxDB a serem distribuídas após a versão 7.5.00. Estes recursos permitirão interoperabilidade entre MaxDB e MySQL:

- Haverá um proxy MySQL permitindo que se conecte ao MaxDb usando o protocolo MySQL. Isto faz com que seja possível usar os programas clientes do MySQL para o MaxDB, com a interface de linha de comando `mysql`, o utilitário de dump `mysqldump`, o programa de importação `mysqlimport`. Usando o `mysqldump`, pode-se facilmente fazer o dump de dados de um sistema de banco de dados e exportar estes dados para outro sistema de banco de dados.
- Replicação entre MySQL e MaxDB será suportado em ambas as direções. Isto é, tanto o MySQL quanto o MaxDb podem ser usados como o servidor master da replicação. O plano a longo prazo é convergir e extender a sintaxe da replicação para que assim ambos os sistemas de bancos de dados entendam a mesma sintaxe. Veja Seção 4.11.1 [Replication Intro], Página 378.

8.6 Links Relacionados ao MaxDB

A página principal para informações sobre o MaxDB é `http://www.mysql.com/maxdb`. Eventualmente, todas as informações disponíveis em `http://www.sapdb.org` serão movidas para lá.

8.7 Palavras Reservadas no MaxDB

Assim como o MySQL, o MaxDB tem algumas palavras reservadas que tenham significados especiais. Normalmente elas não podem ser usadas como nomes de identificadores, tais como nomes de bancos de dados ou tabelas. A tabela a seguir lista as palavras reservadas no MaxDB, indica o contexto no qual estas palavras são utilizadas e indica se elas possuem correspondentes ou não no MySQL. Se existir, o significado no MySQL pode ser idêntico ou diferente em alguns aspectos. O principal objetivo é listar em que o MaxDB difere do MySQL; embora esta lista não esteja completa.

Para a lista de palavras reservadas do MySQL, veja Veja Seção 6.1.7 [Reserved words], Página 479.

Reservada no MaxDB	**Contexto do uso no MaxDB**	**Correspondente no MySQL**
`@`	Pode preceder identificadores, como "`@table`"	Não permitido
`ADDDATE()`	Função SQL	`ADDDATE()`; nova no MySQL 4.1.1
`ADDTIME()`	Função SQL	`ADDTIME()`; nova no MySQL 4.1.1
`ALPHA`	Função SQL	Nenhuma correspondencia
`ARRAY`	Tipo de dados	Não implementado
`ASCII()`	Função SQL	`ASCII()`, mas implementado com um significado diferente
`AUTOCOMMIT`	Transações; `ON` por padrão	Transações; `OFF` por padrão
`BOOLEAN`	Tipos de coluna; `BOOLEAN` aceita como valor apenas `TRUE`, `FALSE`, e `NULL`	`BOOLEAN` was added in MySQL version 4.1.0; it is a synonym for `BOOL` which is mapped to `TINYINT(1)`. It accepts integer values in the same range as `TINYINT` as well as `NULL`. `TRUE` and `FALSE` can be used as aliases for `1` and `0`.
`CHECK`	`CHECK TABLE`	`CHECK TABLE`; similar, mas com uso diferente
`COLUMN`	Tipos de coluna	`COLUMN`; noise word
`CHAR()`	Função SQL	`CHAR()`; identical syntax; similar, not identical usage
`COMMIT`	Implicit commits of transactions happen when data definition queries are being issued	Implicit commits of transactions happen when data definition queries are being issued, but also with a number of other queries
`COSH()`	Função SQL	Nenhuma correspondencia
`COT()`	Função SQL	`COT()`; identical syntax and implementation

CREATE	SQL, data definition language	CREATE
DATABASE	Função SQL	DATABASE(); DATABASE is used in a different context, for example CREATE DATABASE
DATE()	Função SQL	CURRENT_DATE
DATEDIFF()	Função SQL	DATEDIFF(); nova no MySQL 4.1.1
DAY()	Função SQL	Nenhuma correspondencia
DAYOFWEEK()	Função SQL	DAYOFWEEK(); the first day (1) by default is Monday in MaxDB, and Sunday in MySQL
DISTINCT	Funções SQL AVG, MAX, MIN, SUM	DISTINCT; but used in a different context: SELECT DISTINCT
DROP	inter alia in DROP INDEX	DROP INDEX; similar, but not identical usage
EBCDIC()	Função SQL	Nenhuma correspondencia
EXPAND()	Função SQL	Nenhuma correspondencia
EXPLAIN	Optimization	EXPLAIN; similar, but not identical usage
FIXED()	Função SQL	Nenhuma correspondencia
FLOAT()	Função SQL	Nenhuma correspondencia
HEX()	Função SQL	HEX(); similar, but not identical usage
INDEX()	Função SQL	INSTR() or LOCATE(); similar, but not identical syntaxes and meanings
INDEX	USE INDEX, IGNORE INDEX and similar hints are being used right after SELECT, like SELECT ... USE INDEX	USE INDEX, IGNORE INDEX and similar hints are being used in the FROM clause of a SELECT query, like in SELECT ... FROM ... USE INDEX
INITCAP()	Função SQL	Nenhuma correspondencia
LENGTH()	Função SQL	LENGTH(); identical syntax, but slightly different implementation
LFILL()	Função SQL	Nenhuma correspondencia
LIKE	Comparisons	LIKE; but the extended LIKE MaxDB provides rather resembles the MySQL REGEX
LIKE wildcards	MaxDB supports "%", "_", "ctrl+underline", "ctrl+up arrow", "*", and "?" as wildcards in a LIKE comparison	MySQL supports "%", and "_" as wildcards in a LIKE comparison
LPAD()	Função SQL	LPAD(); slightly different implementation
LTRIM()	Função SQL	LTRIM(); slightly different implementation
MAKEDATE()	Função SQL	MAKEDATE(); nova no MySQL 4.1.1
MAKETIME()	Função SQL	MAKETIME(); nova no MySQL 4.1.1
MAPCHAR()	Função SQL	Nenhuma correspondencia
MICROSECOND()	Função SQL	MICROSECOND(); nova no MySQL 4.1.1
NOROUND()	Função SQL	Nenhuma correspondencia
NULL	Column types; comparisons	NULL; MaxDB supports special NULL values that are returned by arithmetic operations that lead to an overflow or a division by zero; MySQL does not support such special values

PI	Função SQL	PI(); identical syntax and implementation, but parantheses are mandatory
REF	Data type	Nenhuma correspondencia
RFILL()	Função SQL	Nenhuma correspondencia
ROWNO	Predicate in WHERE clause	Similar to LIMIT clause
RPAD()	Função SQL	RPAD(); slightly different implementation
RTRIM()	Função SQL	RTRIM(); slightly different implementation
SEQUENCE	CREATE SEQUENCE, DROP SEQUENCE	AUTO_INCREMENT; similar concept, but differing implementation
SINH()	Função SQL	Nenhuma correspondencia
SOUNDS()	Função SQL	SOUNDEX(); slightly different syntax
STATISTICS	UPDATE STATISTICS	ANALYZE; similar concept, but differing implementation
SUBSTR()	Função SQL	SUBSTRING(); slightly different implementation
SUBTIME()	Função SQL	SUBTIME(); nova no MySQL 4.1.1
SYNONYM	Data definition language: CREATE [PUBLIC] SYNONYM, RENAME SYNONYM, DROP SYNONYM	Nenhuma correspondencia
TANH()	Função SQL	Nenhuma correspondencia
TIME()	Função SQL	CURRENT_TIME
TIMEDIFF()	Função SQL	TIMEDIFF(); nova no MySQL 4.1.1
TIMESTAMP()	Função SQL	TIMESTAMP(); nova no MySQL 4.1.1
TIMESTAMP() as argument to DAYOFMONTH() and DAYOFYEAR()	Função SQL	Nenhuma correspondencia
TIMEZONE()	Função SQL	Nenhuma correspondencia
TRANSACTION()	Returns the ID of the current transaction	Nenhuma correspondencia
TRANSLATE()	Função SQL	REPLACE(); identical syntax and implementation
TRIM()	Função SQL	TRIM(); slightly different implementation
TRUNC()	Função SQL	TRUNCATE(); slightly different syntax and implementation
USE	mysql commandline user interface command	USE
USER	Função SQL	USER(); identical syntax, but slightly different implementation, and parantheses are mandatory
UTC_DIFF()	Função SQL	UTC_DATE(); provides a means to calculate the result of UTC_DIFF()
VALUE()	Função SQL, alias for COALESCE()	COALESCE(); identical syntax and implementation
VARIANCE()	Função SQL	Nenhuma correspondencia
WEEKOFYEAR()	Função SQL	WEEKOFYEAR(); nova no MySQL 4.1.1

8.8 Funções

8.9 Tipos de Colunas

9 Conjunto de Caracteres Nacionais e Unicode

Melhora do tratamento dos conjuntos de caracteres é um do recursos adicionado ao MySQL na versão 4.1. Este capítulo explica:

- O que são conjuntos de caracteres e collations
- O sistema padrão de multi níveis
- A nova sintaxe no MySQL 4.1
- Funções e operações afetadas
- O signifcado individual de cada conjunto de caracter e collation

Os recursos descritos aqui estão como implementados no MySQL 4.1.1. (MySQL 4.1.0 possui alguns, mas não todos destes recuros, e alguns deles estão implementados de forma diferente.)

9.1 Conjuntos de Caracteres e Collations em Geral

Um **conjunto de caracters** é um conjunto de simbolos e códigos. Uma **collation** é um conjunto de regras para comparação de caracteres em um conjunto de caracteres. Vamos deixar a distinção clara com um exemplo de um conjunto de caracteres imaginário.

Suponha que temos um alfabeto com quatro letras: '`A`', '`B`', '`a`', '`b`'. Damos um número a cada letra: '`A`' = 0, '`B`' = 1, '`a`' = 2, '`c`' = 3. A letra '`A`' é o símbolo, o número 0 é o **código** para '`A`', e a combinação de todas as quatro letra e seus códigos é um **conjunto de caracteres**.

Agora suponha que desejamos comparar dusa strings, '`A`' e '`B`'. O modo mais simples de se fazer isto é olhar o código — 0 para '`A`' e 1 para '`B`' — e como 0 é menor que 1, dezemos que '`A`' é menor que '`B`'. Agora, o que fizemos foi apenas aplicar um collation a nosso conjunto de caracteres. A collation é um conjunto de regras (apenas um regra neste caso): "compara os códigos". Chamamos isto a mais simples de todas as collations possíveis como um collation `binária`.

Mas e se você dissesse que letras mínusculas e maiúsculas são equivalentes? Então haveriam pelo menos duas regras: (1) tratar as letras minúsculas '`a`' e '`b`' como equivalentes a '`A`' e '`B`'; (2) e então comparar os códigos. Chamamos isto de collation **caso insensitivo**. É um pouco mais complexo do que collation binária.

Na vida real, a maioria dos conjuntos de caracteres possuem muitos caracteres: não apenas '`A`' e '`B`' mas todo o alfabeto, algumas vezes alfabetos múltiplos ou sistemas de escritas ocidentais com milhares de caracteres, junto com muitos símbolos especiais e sinais de pontuação. Em geral as collations também possuem diversas regras: não apenas caso insensitivo mas acentos insensitivos e mapeamento de múltiplos caracteres (como a regra de que '`Ö`' = '`OE`' em uma das duas collations alemãs).

O MySQL 4.1 pode fazer as seguintes coisas para você:

- Armazena a string usando um variedade de conjunto de caracteres
- Compara strings usando uma variedade de collations
- Mistura strings com diferentes conjuntos de caracteres ou collations no mesmo servidor, o mesmo banco de dados ou a mesma tabela

- Permite a especificação de conjunto de caracteres e collations em qualquer nível

A este respeito, o MySQL 4.1 não só é mais flexível que o MySQL 4.0, mas também está bem a frente de outros SGBDs. No entanto, para usar os novos recursos efetivamente, você precisará aprender quais conjuntos de caracteres e collations estão disponíveis, como alterar os seus padrões e o que os vários operadores de string fazem como ele.

9.2 Conjunto de Caracteres e Collations no MySQL

Um conjunto de caracter sempre tem pelo menos uma collation. Ele pode ter diversas collations.

Por exemplo, conjunto de caracteres `latin1` ("ISO-8859-1 West European") tem os seguintes collations:

Collation	**Significado**
`latin1_bin`	Binario de acordo com a codificação `latin1`
`latin1_danish_ci`	Dinamarquês/Norueguês
`latin1_german1_ci`	Alemão DIN-1
`latin1_german2_ci`	Alemão DIN-2
`latin1_swedish_ci`	Sueco/Finnish
`latin1_general_ci`	Multilingua

Notas:

- Dois conjuntos de caracteres diferentes não podem ter a mesma collation.
- Cada conjunto de caracteres tem uma collation que é a *collation padrão*. Por exemplp, o collation padrão para `latin1` é `latin1_swedish_ci`.

Perceba que existe uma convenção para nomes de collations: Elas iniciam com o nome do conjunto de caracteres com o qual elas são associadas, eles normalmente incluem um nome de linguagem e finalizam com `_ci` (caso insensitivo), `_cs` (caso sensitivo), ou `_bin` (binario).

9.3 Determinando o Conjunto de Caracteres e Collation Padrões

Existem configurações padrões para conjuntos de caracteres e collations em quatro níveis: servidor, banco de dados, tabela, conexão. A seguinte descrição pode parecer complexa, mas será encontrada na prática que os padrões em multi-níveis levam a resultados naturais e óbvios.

9.3.1 Conjunto de Caracteres e Collations do Servidor

O MySQL Server possui um conjunto de caracteres de servidor e collation de servidor que não podem ser nulos.

O MySQL determina o conjunto de caracteres e collations de servidor desta forma:

- De acordo com as opções de configuração em efeito quando o servidor é iniciado.

Neste nível, a decisão é simples. O conjunto de caracteres e collations do servidor dependem das opções que você usa quando você inicia o `mysqld`. Você pode usar `--default-character-set=character_set_name` para o conjunto de caracteres, e junto com isto você pode adcionar `--default-collation=collation_name` para a collation. Se você não especificar um conjunto de caracteres, é o mesmo que utilizar `--default-character-set=latin1`. Se você especificar apenas um conjunto de caracteres (por exemplo, `latin1`) mas não uma collation, é o mesmo que usar `--default-charset=latin1 --collation=latin1_swedish_ci` pois `latin1_swedish_ci` é a collation padrão para `latin1`. Desta forma, os três comando seguintees todos têm o mesmo efeito:

```
shell> mysqld
shell> mysqld --default-character-set=latin1
shell> mysqld --default-character-set=latin1
           --default-collation=latin1_swedish_ci
```

Um modo de o conjunto é recompilando. Se você quiser alterar o conjunto de caracteres e collation padrões na construção dos fontes, utilize: `--with-character-set` e `--with-collation` como argumento para `configure`. Por exemplo:

```
shell> ./configure --with-character-set=latin1
```

ou

```
shell> ./configure --with-character-set=latin1
           --with-collation=latin1_german1_ci
```

Tanto o `mysqld` quanto o `configure` verificam que a combinação conjunto de caracteres/collations é válida. Cada programa exibe um mensagem de erro e termina se a combinação não for válida.

9.3.2 Conjunto de Caracteres e Collation de Banco de Dados

Todo banco de dados tem um conjunto de caracteres de banco de dados e uma collatio de banco de dados, que não podem ser nulos. Os comandos `CREATE DATABASE` e `ALTER DATABASE` agora possuem cláusulas opcionais para especificarem o collation e conjunto de caracteres de banco de dados:

```
CREATE DATABASE db_name
   [DEFAULT CHARACTER SET character_set_name [COLLATE collation_name]]

ALTER DATABASE db_name
    [DEFAULT CHARACTER SET character_set_name [COLLATE collation_name]]
```

Exemplo:

```
CREATE DATABASE db_name
   DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
```

O MySQL escolhe o conjunto de caracteres e collations do banco de dados desta forma:

- Se `CHARACTER SET X` e `COLLATE Y` foram especificados, então o conjunto de caracteres é `X` e a é collation `Y`.
- Se `CHARACTER SET X` foi especificado sem `COLLATE`, então o conjunto de caracteres é `X` e a collation é o padrão.

- Senão utiliza o conjunto de caracteres e a collation de servidor.

A sintaxe `CREATE DATABASE ... DEFAULT CHARACTER SET ...` do MySQL é análoga a sintaxe `CREATE SCHEMA ... CHARACTER SET ...` do padrão SQL. Por isto, é possível criar bancos de dados com com conjunto de caracteres e collations diferentes, no mesmo servidor MySQL.

O conjuto de caracteres e collations do banco de dados são usados como valores padrões se o conjunto de caracteres e a collation de tabela não forem especificados nas instruções `CREATE TABLE`. Eles não possuem nenhum outro propósito.

9.3.3 O Conjunto de Caracteres e Collations de Tabela

Toda tabela tem um conjunto de caracteres e collations de tabela, que não pode ser nulo. As instruções `CREATE TABLE` e `ALTER TABLE` agora possuem um cláusula opcional para especificar o conjunto de caracteres e collation de tabela:

```
CREATE TABLE table_name ( column_list )
   [DEFAULT CHARACTER SET character_set_name [COLLATE collation_name]]

ALTER TABLE table_name
   [DEFAULT CHARACTER SET character_set_name] [COLLATE collation_name]
```

Exemplo:

```
CREATE TABLE t1 ( ... ) DEFAULT CHARACTER SET latin1 COLLATE latin1_danish_ci;
```

O MySQL escolhe o conjunto de caracteres e collation de tabela desta forma:

- Se `CHARACTER SET X` e `COLLATE Y` forem especificados, então o conjunto de caracteres é `X` e collation é `Y`.
- Se `CHARACTER SET X` foi especificado sem `COLLATE`, então o conjunto de caracteres é `X` e o collation é o padrão.
- Senão, o conjunto de caracteres e collation são os padrões.

O conjunto de caracteres e collation de tabela são usado como valores padrões, se o conjunto de caracteres e collation de colunas não são especificados nas definições de colunas individuais. O conjunto de caracteres e collation de tabelas são extensões MySQL; não há nada deste tipo na padrão SQL.

9.3.4 Conjunto de Caracteres e Collation de Colunas

Toda coluna "caracter" (isto é, uma colua do tipo `CHAR`, `VARCHAR`, ou `TEXT`) tem um conjunto de caracteres e collation de coluna, que não pode ser nulo. A sintaxe de definição de coluna agora possui uma cláusula opcional para especificar o conjunto de caracteres e collation:

```
column_name {CHAR | VARCHAR | TEXT} (column_length)
    [CHARACTER SET character_set_name [COLLATE collation_name]]
```

Exemplo:

```
CREATE TABLE Table1
(
   column1 VARCHAR(5) CHARACTER SET latin1 COLLATE latin1_german1_ci
```

```
);
```

O MySQL escolhe o conjunto de caracteres e collation de coluna desta forma:

- Se `CHARACTER SET X` e `COLLATE Y` forem especificados, então o conjunto de caracteres é `X` e collation é `Y`.
- Se `CHARACTER SET X` foi especificado sem `COLLATE`, então o conjunto de caracteres é `X` e o collation é o padrão.
- Senão, o conjunto de caracteres e collation são os padrões.

As cláusulas `CHARACTER SET` e `COLLATE` são do padrão SQL.

9.3.5 Exemplos de Atribuições de Conjuntos de Caracteres e Collation

Os seguintes exemplos mostram como o MySQL determina valores de conjunto de caracteres e collations padrões.

Exemplo 1: Definição de Tabela + Coluna

```
CREATE TABLE t1
(
  c1 CHAR(10) CHARACTER SET latin1 COLLATE latin1_german1_ci
) DEFAULT CHARACTER SET latin2 COLLATE latin2_bin;
```

Aqui você tem uma coluna com um conjunto de caracteres `latin1` e um collation `latin1_german1_ci`. A definição é explicita, assim ele é direto. Note que não há problemas em armazenar uma coluna `latin1` em uma tabela `latin2`.

Example 2: Definição de Tabela + Coluna

```
CREATE TABLE t1
(
   c1 CHAR(10) CHARACTER SET latin1
) DEFAULT CHARACTER SET latin1 COLLATE latin1_danish_ci;
```

Desta vez temos uma coluna com um conjunto de caracteres `latin1` e uma collation padrão. Agora, embora possa parecer natural, a collation padrão é tomada do nível de tabela. Como a collation padrão para `latin1` é sempre `latin1_swedish_ci`, a coluna `c1` terá uma collation `latin1_swedish_ci` (e não `latin1_danish_ci`).

Exemplo 3: Definição de Tabela + Coluna

```
CREATE TABLE t1
(
   c1 CHAR(10)
) DEFAULT CHARACTER SET latin1 COLLATE latin1_danish_ci;
```

Temos uma coluna com um conjunto de caracteres padrão e uma collation padrão. Nesta circunstância, o MySQL olha para o nível de tabela para determinar o conjunto de caracteres e collation de coluna. Assim o conjunto de caracteres para colune `c1` é `latin1` e sua collation é `latin1_danish_ci`.

Exemplo 4: Definição de Banco de Dados + Tabela + Coluna

```
CREATE DATABASE d1 DEFAULT CHARACTER SET latin2 COLLATE latin2_czech_ci;
USE d1;
CREATE TABLE t1
(
   c1 CHAR(10)
);
```

Criamos uma coluna sem especificar seu conjunto de caracteres e collation. Também não especificamos um conjunto de caracteres e uma collation na nível de tabela. Nestas circubntâncias, o MySQL olha para o nível de banco de dados para a determinação. (A configuração do banco de dados se torna a configuração da tabela e então a configuração da coluna). Assim o conjunto de caracteres para coluna `c1` é `latin2` e sua collation é `latin2_czech_ci`.

9.3.6 Conjunto de Caracteres e Collation de Conexão

Toda conexão tem o seu conjunto de caracteres e collation, que não podem ser nulos. Esistem atualmente dois conjuntos de caracteres de conexão, que chamamos "connection/literals" e "connection/results" quando é necessário distingui-los.

Considere o que é uma "conexão": é o que você faz quando conecta ao servidor. O cliente envia instruções SQL, como consultas, pela conexão com o sevidor. O servidor envia respostas, como resultados, pela conexão de volta para o cliente. Isto leva a diversas questões, tal como: (a) em qual conjunto de caracteres está uma consulta quando ela deixa o cliente? (b) em qual conjunto de caracteres o servidor deve traduzir uma consulta após recebê-la? (c) para qual conjunto de caracteres o servidor deve traduzir antes de enviar o resultado ou mensagem de erros de volta para o cliente? Você pode fazer um ajuste fino das configurações para isto, ou você pode depender dos padrões (neste caso, você pode ignorar esta seção).

Existem suas instruções que afetam o conjunto de caracteres da conexão:

```
SET NAMES character_set_name
SET CHARACTER SET character_set_name
```

`SET NAMES` indica o que está na instrução SQL que o cliente envia. Assim, `SET NAMES cp1251`diz ao servidor que "futuras mensagens vindas do cliente estarão no conjunto de caracteres `cp1251`" e o servidor está livre para traduzir para seu próprio conjunto de caracteres, se apropriado.

`SET CHARACTER SET` indica o que está na instrução SQL que o cliente envia, e também o que está no resultado que o servidor envia de volta para o cliente. Assim, `SET CHARACTER SET` inclui `SET NAMES`, e também especifica qual conjunto de caracteres o valor da coluna terá se, por exempo, você usar uma instrução `SELECT`.

EXEMPLO: Suponha que `column1` é definido como `CHAR(5) CHARACTER SET latin2`. Se você não utilizar `SET CHARACTER SET`, então para `SELECT column1 FROM t` o servidor enviará de volta todos os valores para `column1` usando o conjunto de caracteres `latin2`. Se por outro lado você usar `SET CHARACTER SET latin1` então o servidor, antes de enviar de volta, converterá os valores `latin2` para `latin1`. Tal conversão é lenta e poder ter perdas.

Quando você executa `SET NAMES` ou `SET CHARACTER SET`, você também está alterando a "collation da conexão". No entanto a collation da conexão existe apenas para consistência. Normalmente o seu valor não importa.

Com o cliente `mysql`, não é necessário executar `SET NAMES` todas as vezes que você inicá-lo. Você pode adicionar a opção `--default-character-set-name` a sua linha de instrução do `mysql`, ou em seu arquivo de opção. Por exemplo, a seguinte configuração do arquivo de opção irá alterar o conjunto de caracteres da conexão cada vez que você executar `mysql`:

```
[mysql]
default-character-set-name=character_set_name
```

9.3.7 Conjunto de Caracteres e Collation de Caracter de String Literal

Todo caracter de uma string literal tem um conjunto de caracteres e collation, que podem ser nulos.

Um caracter de uma string literal pode ter um introdutor de conjunto de caracteres opcional e cláusula `COLLATE`:

```
[_character_set_name]'string' [COLLATE collation_name]
```

Exemplos:

```
SELECT 'string';
SELECT _latin1'string';
SELECT _latin1'string' COLLATE latin1_danish_ci;
```

A instrução simples `SELECT 'string'` usa o conjunto de caracteres da conexão/literal.

A expressão `_character_set_name` é formalmente chamada um *introdutor*. Ele diz ao analisador que "a string que ele vai seguir está no conjunto de caracteres `X`." Como isto tem confundido as pessoas no passado, enfatizamos que um introdutor não faz qualquer conversão, ele simplesmente um sinal que não altera o valor da string. Um introdutor também é permitido antes de uma notação de um literal hexa padrão e um literal hexa numérico (`x'literal'` e `0xnnnn`), e antes de `?` (substituição de parâmetros ao usar intruções preparadas dentro de uma interface de linguagem de programação).

Exemplos:

```
SELECT _latin1 x'AABBCC';
SELECT _latin1 0xAABBCC;
SELECT _latin1 ?;
```

O MySQL determina um conjunto de caracteres e collation de literal desta forma:

- Se `_X` e `COLLATE Y` forma especificados então o conjunto de caracteres do literal é `X` e o collation do literal é `Y`
- Se `_X` é especificado mas `COLLATE` não é especificado, então o conjunto de caracteres do literal é `X` e a collation do literal é a collation padrão do `X`
- De outra forma, o conjunto de caracteres e collation é o da conexão/literal.

Exemplos:

- Uma string com o conjunto de caracteres `latin1` e collation `latin1_german1_ci`.

  ```
  SELECT _latin1'Müller' COLLATE latin1_german1_ci;
  ```

- Uma string com conjunto de caracteres `latin1` e e sua collation padrão, isto é, `latin1_swedish_ci`:

```
SELECT _latin1'Müller';
```

- Uma string com o conjunto de caracteres e a collation da conexão/literal:

```
SELECT 'Müller';
```

Introdutores de conjunto de caracteres e a cláusula `COLLATE` são implementados de acordo com as especificações do padrão SQL.

9.3.8 Cláusula `COLLATE` em Várias Partes de uma Consulta SQL

Com a cláusula `COLLATE` você pode sobrescrever o padrão da collation, qualquer que seja ele, para comparação. `COLLATE` pode ser usada em várias partes da consulta SQL. Aqui estão alguns exemplos:

- Com `ORDER BY`:

```
SELECT k
FROM t1
ORDER BY k COLLATE latin1_german2_ci;
```

- Com `AS`:

```
SELECT k COLLATE latin1_german2_ci AS k1
FROM t1
ORDER BY k1;
```

- Com `GROUP BY`:

```
SELECT k
FROM t1
GROUP BY k COLLATE latin1_german2_ci;
```

- Com aggregate functions:

```
SELECT MAX(k COLLATE latin1_german2_ci)
FROM t1;
```

- Com `DISTINCT`:

```
SELECT DISTINCT k COLLATE latin1_german2_ci
FROM t1;
```

- Com `WHERE`:

```
SELECT *
FROM t1
WHERE _latin1 'Müller' COLLATE latin1_german2_ci = k;
```

- Com `HAVING`:

```
SELECT k
FROM t1
GROUP BY k
HAVING k = _latin1 'Müller' COLLATE latin1_german2_ci;
```

9.3.9 Precedência da Cláusula COLLATE

A cláusula `COLLATE` tem alta precedência (maior que `||`), então a expressão

```
x || y COLLATE z
```

é equivalente a:

```
x || (y COLLATE z)
```

9.3.10 Operador BINARY

O operador `BINARY` é uma atalho para uma cláusula `COLLATE`. Por exemplo, `BINARY 'x'` é equivalente a `'x' COLLATE y`, onde `y` é o nome de uma collation binária apropriada. Por exemplo, assumindo que a coluna `a` é do conjunto de caracteres `latin1`, estas duas consultas têm o mesmo efeito:

```
SELECT * FROM t1 ORDER BY BINARY a;
SELECT * FROM t1 ORDER BY a COLLATE latin1_bin;
```

Nota: Todo conjunto de caracteres tem um collation binário.

9.3.11 Alguns Casos Especiais Onde a Determinação da Collation e Trabalhosa

Na grande maioria das consultas, é obvio qual collation que o MySQL usa para resolver uma operação de comparação. Por exemplo, nos seguintes casos deve estar claro que a collationserá "a collation de coluna da coluna `x`":

```
SELECT x FROM T ORDER BY x;
SELECT x FROM T WHERE x = x;
SELECT DISTINCT x FROM T;
```

No entanto, quando múltiplos operandos estão envolvidos, pode haver ambiguidade. Por exemplo:

```
SELECT x FROM T WHERE x = 'Y';
```

Esta consulta deve usar a collation de coluna `x`, ou da string literal `'Y'`?

O padrão SQL resolve tal questão usando o que se costuma chamar real "coercibilidade". A essência é: Como `x` e `'Y'` tem collation, qual collation toma precedência? É complexo, mas estas regras cuidariam da maioria das situações:

- Uma cláusula `COLLATE` explicita tem precedência 4
- Uma concatenação de duas strings com diferentes collations tem precedência 3.
- Uma collation de coluna tem precedência 2.
- Uma collation de literal tem precedência 1.

Estas regras resolvem ambiguidades da seguinte forma:

- Use a collation com a maior precedência.
- Se ambos os lados tiverem a mesma precedência, então terá um erro se a collation não são as mesmas.

Exemplos:

`column1 = 'A'`	Usa a collation de `column1`
`column1 = 'A' COLLATE x`	Usa a collation de `'A'`
`column1 COLLATE x = 'A' COLLATE y`	Error

9.3.12 Collations Devem Ser para o Conjunto de Caracteres Certo

Lembramos que cada conjunto de caracteres tem um ou mais collation, e cada collation é associada com um e apenas um conjunto de caracteres. Consequentemente, a seguinte instrução causa um mensagem de erro porque a collation `latin2_bin` não é permitida com o conjunto de caracteres `latin1`:

```
mysql> SELECT _latin1 'x' COLLATE latin2_bin;
ERROR 1251: COLLATION 'latin2_bin' is not valid
for CHARACTER SET 'latin1'
```

9.3.13 Um exemplo do Efeito da Collation

Suponha que a coluna `X` na tabela `T` possui estes valores na coluna `latin1`:

```
Muffler
Müller
MX Systems
MySQL
```

E suponha que os valores da coluna são retornados usando a seguinte instrução:

```
SELECT X FROM T ORDER BY X COLLATE collation_name;
```

A ordem resultante dos valores para diferentes collation é mostrado nesta tabela:

`latin1_swedish_ci`	`latin1_german1_ci`	`latin1_german2_ci`
Muffler	Muffler	Müller
MX Systems	Müller	Muffler
Müller	MX Systems	MX Systems
MySQL	MySQL	MySQL

A tabela é um exemplo que mostra que mostra qual seria o efeito se usassemos collation diferentes em um cláusula `ORDER BY`. O caracter que está causando o problema neste exemplo é o U com dois pontos sobre ele, que os Alemães chamam de U-umlaut, mas nós chamamos de U-diaeresis.

A primeira coluna mostra o resultado da `SELECT` usando as regras de collation Suéco/Finlandês, que diz que U-diaeresis ordena com Y.

A segunda coluna mostra o resultado da `SELECT` usando as regras Almão DIN-1, que diz que U-diaeresis ordena com U.

A terceira coluna mostra o resultado da `SELECT` usando as regras Almão DIN-2, que diz que U-diaeresis ordena com UE.

Três collation diferentes, três resultados diferentes. Isto é o que o MySQL está aqui para tratar. Usando a collation apropriada, você pode esclher a ordem que você deseja.

9.4 Operações Afetadas pelo Suporte a Conjunto de Caracteres

Esta seção descreve operacões que pegam a informação do conjunto de caracteres dentro da conta agora.

9.4.1 Strings de Resultados

O MySQL tem muitos operadores e funções que retornam um string. Esta seção responde a questão: Qual é o conjunto de caracteres e collation de um certa string?

Para funções simples que pegam uma string de entrada e retornam uma string de resultado como saída, a saída do conjunto de caracteres e collation são as mesmas da entrada principal. Por exemplo, `UPPER(X)` retorna uma string cuja string de caracter e collation são os mesmo de `X`. O mesmo se aplica a: `INSTR()`, `LCASE()`, `LOWER()`, `LTRIM()`, `MID()`, `REPEAT()`, `REPLACE()`, `REVERSE()`, `RIGHT()`, `RPAD()`, `RTRIM()`, `SOUNDEX()`, `SUBSTRING()`, `TRIM()`, `UCASE()`, `UPPER()`. (Note também: a função `REPLACE()`, diferente de todas as outras funções, ignora a collation da string de entrada e realiza uma comparação de caso-insensitivo todas as vezes.)

Para operações que combinam múltiplas entradas de string e retornam uma única saída de string, As "regras de agregamento" do SQL-99 se aplicam. Eles são:

- Se ocorrer um `COLLATE X` explicito, então use `X`
- Se ocorrerem `COLLATE X` e `COLLATE Y` explicitos, então erro
- Senão, se todas as collations são `X`, então use `X`
- Senão, o resultado não possui collation

Por exemplo, com `CASE ... WHEN a THEN b WHEN b THEN c COLLATE X END`, a collation resultante é `X`. O mesmo se aplica a: `CONCAT()`, `GREATEST()`, `IF()`, `LEAST()`, `CASE`, `UNION`, `||`, `ELT()`.

Para operações que convertem para dados de caracteres, o resultado do conjunto de caracteres e collation da string estão no *connection/literals character set* e possuem a *connection/literals collation*. Isto se aplica a: `CHAR()`, `CAST()`, `CONV()`, `FORMAT()`. `HEX()`, `SPACE()`.

9.4.2 `CONVERT()`

`CONVERT()` fornece um modo de converter dados entre diferentes conjunto de caracteres. A sintaxe é:

```
CONVERT(expr USING transcoding_name)
```

No MySQL, nomes transcodificados são o mesmo que o nomes dos conjuntos de caracteres correspondentes.

Exemplos:

```
SELECT CONVERT(_latin1'Müller' USING utf8);
INSERT INTO utf8table (utf8column)
   SELECT CONVERT(latin1field USING utf8) FROM latin1table;
```

`CONVERT(... USING ...)` é implementado de acordo com a especificação SQL-99.

9.4.3 CAST()

Você também pode usar `CAST()` para converter uma string para um conjunto de caracteres diferente. O novo formato é:

```
CAST ( character_string AS character_data_type
    CHARACTER SET character_set_name )
```

Exemplo:

```
SELECT CAST(_latin1'test' AS CHAR CHARACTER SET utf8);
```

Você não usar uma cláusula `COLLATE` dentro de um `CAST()`, mas você pode usá-la fora, isto é, `CAST(... COLLATE ...)` é ilegal mas `CAST(...) COLLATE ...` é permitido.

Exemplo:

```
SELECT CAST(_latin1'test' AS CHAR CHARACTER SET utf8) COLLATE utf8_bin;
```

Se você usar `CAST()` sem especificar `CHARACTER SET`, então o conjunto de caracteres e collation resultante são o conjunto de caracteres da conexão/literal e a sua collation padrão. Se você usar `CAST()` com `CHARACTER SET X`, então o conjunto de caracteres resultante é `X` e a collation resultante é a collation padrão de `X`.

9.4.4 SHOW CHARACTER SET

O comando `SHOW CHARACTER SET` exibe todos os conjunto de caracteres dsiponíveis. Ele aceita uma cláusula `LIKE` opcional que indica qual nome de conjunto de caracteres coincidir.

Por exemplo:

```
mysql> SHOW CHARACTER SET LIKE 'latin%';
+---------+-----------------------------+-------------------+--------+
| Charset | Description                 | Default collation | Maxlen |
+---------+-----------------------------+-------------------+--------+
| latin1  | ISO 8859-1 West European    | latin1_swedish_ci |      1 |
| latin2  | ISO 8859-2 Central European | latin2_general_ci |      1 |
| latin5  | ISO 8859-9 Turkish          | latin5_turkish_ci |      1 |
| latin7  | ISO 8859-13 Baltic          | latin7_general_ci |      1 |
+---------+-----------------------------+-------------------+--------+
4 rows in set (0.00 sec)
```

Notas sobre a lista precedente:

- A coluna `Maxlen` exie o número máximo de bytes usado para armazenar um caracter.

9.4.5 SHOW COLLATION

A saída de `SHOW COLLATION` inclui todos os conjunto de caracteres disponíveis. Ele tem uma cláusula `LIKE` opcional que indice com qual nome de collation que ele deve coincidir.

```
mysql> SHOW COLLATION LIKE 'latin1%';
+-------------------+---------+----+---------+----------+---------+
| Collation         | Charset | Id | Default | Compiled | Sortlen |
+-------------------+---------+----+---------+----------+---------+
```

```
| latin1_german1_ci | latin1  |  5 |         |          |       0 |
| latin1_swedish_ci | latin1  |  8 | Yes     | Yes      |       0 |
| latin1_danish_ci  | latin1  | 15 |         |          |       0 |
| latin1_german2_ci | latin1  | 31 |         | Yes      |       2 |
| latin1_bin        | latin1  | 47 |         | Yes      |       0 |
| latin1_general_ci | latin1  | 48 |         |          |       0 |
| latin1_general_cs | latin1  | 49 |         |          |       0 |
+-------------------+---------+----+---------+----------+---------+
7 rows in set (0.00 sec)
```

A coluna `Default` indica se uma collation é o padrão para o seu conjunto de caracteres. `Compiled` indica se o conjunto de caracteres é ou não compilado no servidor. `Sortlen` é relacionado a quantidade de memória exigida para armazenar strings expressadas no conjunto de caracteres.

9.4.6 SHOW CREATE DATABASE

A consulta seguinte mostra uma instrução `CREATE DATABASE` que criará o banco de dados dado. O resultado inclui todas as opções de banco de dados. `DEFAULT CHARACTER SET` e `COLLATE` são suportados. Todas as opções de banco de dados são armazenadas em um arquivo texto que pode se encontrado no diretório de banco de dados.

```
mysql> SHOW CREATE DATABASE a;
+----------+-------------------------------------------------------------------------
| Database | Create Database
+----------+-------------------------------------------------------------------------
| a        | CREATE DATABASE `a` /*!40100 DEFAULT CHARACTER SET macce COLLATE macce
+----------+-------------------------------------------------------------------------
1 row in set (0.00 sec)
```

9.4.7 SHOW FULL COLUMNS

A instrução `SHOW COLUMNS` agora mostra as collations das colunas da tabela, quando chamado como `SHOW FULL COLUMNS`. Colunas com tipos de dados `CHAR`, `VARCHAR` ou `TEXT` tem collation não-`NULL`. Tipos numéricos e outros que não seja caracteres tem collations `NULL`. Por exemplo:

```
mysql> SHOW FULL COLUMNS FROM a;
+-------+---------+-------------------+------+-----+---------+-------+
| Field | Type    | Collation         | Null | Key | Default | Extra |
+-------+---------+-------------------+------+-----+---------+-------+
| a     | char(1) | latin1_swedish_ci | YES  |     | NULL    |       |
| b     | int(11) | NULL              | YES  |     | NULL    |       |
+-------+---------+-------------------+------+-----+---------+-------+
2 rows in set (0.02 sec)
```

O conjunto de caracteres não é parte do display.

9.5 Suporte Unicode

Existem dois novos conjunto de caracteres para armazenar dados Unicode: `ucs2` (o conjunto de caracteres UCS-2 Unicode) e `utf8` (a codificação UTF-8 do conjunto de caracteres do Unicode).

- Na UCS-2 (representação Unicode binária) todo caracter é representado por um código Unicode de dois bytes com o byte mais significante primeiro. Por exemplo: "LATIN CAPITAL LETTER A" tem o código 0x0041 e é armazenado como uma sequência de dois bytes: 0x00 0x41. "CYRILLIC SMALL LETTER YERU" (Unicode 0x044B) é armazenada como uma sequência de dois bytes: 0x04 0x4B. Para caracteres Unicode e seus código veja a Unicode Home Page (`http://www.unicode.org/`).

  Restrição temporária: UCS-2 não pode (ainda) ser usado como um conjunto de caracteres de cliente. Insto significa que `SET NAMES ucs2` não funcionará.
- O conjunto de caracteres UTF8 (representação Unicode trasnformada) é um modo alternativo de armazenar dados Unicode. Ele é implementado de acordo com a RFC2279. A idéia do conjunto de caracteres UTF8 é que vários caracteres Unicodes cobem em uma sequência de bytes de tamanhos diferentes.
  - Letras, digitos e sinais de pontuação do Latin básico usam um byte.
  - A maioria das letras script da Europa e Oriente Médio cabem em uma sequência de dois bytes: letras Latin extendidas (com til, agudo, grave e outros acentos), Cirílico, Grego, Armenio, Hebreu, Árabe, Sírio e outors.
  - Ideógrafos Coreanos, Chineses e Japoneses usam sequências de três bytes.
  - Atualmente, o suporte MySQL UTF8 não inclui sequências de quatro-bytes.

  Dica: economize spaço com UTF8, use `VARCHAR` em vez de `CHAR`. Senão, o MySQL tem que reservar 30 bytes para uma coluna `CHAR(10) CHARACTER SET utf8`, pois este é o tamanho máximo possível.

9.6 UTF8 para Metdados

O metadados é o dado sobre o dado. Qualquer coisa que descreva os bancos de dados, como o opsto de ser o conteúdo do banco de dados, é metadados. Assim nomes de colunas, banco de dados, usuários, versões e a maioria dos resultados strings de `SHOW`, são metadados.

Todos os metadados devem estar no mesmo conjunto de caracteres. (Senão, `SHOW` não funcionaria corretamente devido aos diferentes registros na mesma coluna estarem em conjunto de caracteres diferentes). Por outro lado, metadados devem incluir todos os caracteres em todas as linguagens (senào os usuários não poderiam nomear as colunas e tabelas na suas próprias linguagens). Para permitir ambos os objetivos, o MySQL armazena metadados em um conjunto de caracteres Unicode, chamado UTF8. Isto não causa qualquer rompimento se você nunca usar caracteres acentuados. Mas se você fizer, deverá estar ciente que o metadado está em UTF8.

Isto significa que funções `USER()` (e seus sinônimos), `SESSION_USER()` and `SYSTEM_USER()`), `CURRENT_USER()`, e `VERSION()` terá o conjunto de caracteres UTF8 por padrão.

Isto NÃO significa que o cabeçalho das colunas e os resultados da função `DESCRIBE` estarão no conjunto de caracteres UTF8 por padrão. (Quando você fizer `SELECT column1 FROM t` o

nome `column1` será retornado do servidor para o cliente no conjunto de caracteres do cliente como determinado pela instrução `SET NAMES`.)

Se você quizer que o servidor passe o resultado de volta em um conjunto de caracteres não-UTF8, então use `SET CHARACTER SET` para forçar o servidor a converter (veja Seção 9.3.6 [Charset-connection], Página 715), ou configurar o cliente para fazer a a conversão, mas esta opção não estará disponível para muitos clientes até no final no ciclo do produto MySQL 4.x.

Se você está apenas usando, por exemplo, a função `USER()` para comparação ou atribuição dentro de uma única instrução ... não preocupe. O MySQL fará alguma conversão automática para você.

```
SELECT * FROM Table1 WHERE USER() = latin1_column;
```

Isto funcionará, porque o conteúdo de `latin1_column` é convertido automaticamente para UTF8 antes da comparação.

```
INSERT INTO Table1 (latin1_column) SELECT USER();
```

Isto funcionará, porque o contéudo de `USER()` é convertido automaticamente para `latin1` antes da atribuição. A conversão automática ainda não está totalmente implementada, mas deve funcionar corretamente em uma versão posterior.

Embora a conversão automática não esteja no padrão SQL, o documento do padrão SQL diz que todo conjunto de caracteres é (em termos de caracteres suportados) um "subconjunto" do Unicode. Desde que isto seja um princípio bem conhecido que "o que aplica a um superconjunto pode ser aplicado a um subconjunto", acreditamos que uma collation para Unicode pode ser aplicado para comparações com strings não -Unicode.

NATA DA VERSÃO 4.1.1: Os arquivos '`errmsg.txt`' estarão todos em UTF8 depois deste ponto. Conversão o conjunto de caracteres do clientes serão automáticos, como para metadados. Também: Podemos alterar o comportamento padrão para passar de volta o metadado do resultado em um futuro próximo.

9.7 Compatibilidade com Outros SGBDs

Para compatibilidade com o SAP DB estas duas instruções são a mesma:

```
CREATE TABLE t1 (f1 CHAR(n) UNICODE);
CREATE TABLE t1 (f1 CHAR(n) CHARACTER SET ucs2);
```

9.8 Novo Formato do Arquivo de Configuração do Conjunto de Caracteres

No MySQL 4.1, a configuração de um conjunto de caracteres é armazenado em um arquivo XML, um arquivo por conjunto de caracteres (na versão anterior, esta informação era armazenada em arquivos '`.conf`')

9.9 Conjunto de Caracteres Nacional

No MySQL-4.x e mais novos, `NCHAR` e `CHAR` eram sinônimos. ANSI define `NCHAR` ou `NATIONAL CHAR` como um modo de definir que uma coluna `CHAR` deve usar alguns conjuntos de caracteres predefinidos. O MySQL usa `utf8` como o conjunto de caracteres predefinido. Por exemplo, estas declarações de tipos de colunas são equivalentes:

```
CHAR(10) CHARACTER SET utf8
NATIONAL CHARACTER(10)
NCHAR(10)
```

Como estas:

```
VARCHAR(10) CHARACTER SET utf8
NATIONAL VARCHAR(10)
NCHAR VARCHAR(10)
NATIONAL CHARACTER VARYING(10)
NATIONAL CHAR VARYING(10)
```

Você pode usar `N'literal'` para criar uma string em um conjunto de caracteres nacional.

Estas duas instruções são equivaletes:

```
SELECT N'some text';
SELECT _utf8'some text';
```

9.10 Atualizando para o MySQL 4.0

Agora, e sobre a atalização de versões mais antigas do MySQL? o MySQL 4.1 é quase compaível com o MySQL 4.0 e versões anteriores pela simples razão que quase todos os recursos são novos, então não há nada em versões anteriores que conflitem com ele. No entanto, existem algumas diferenças e poucas coisas com as quais deve estar ciente.

O mais importante: O "conjunto de caracteres do MySQL 4.0" tem as propriedades do "conjunto de caracteres do MySQL 4.1" e da "collation do MySQL 4.1". Você terá que desaprender isto. qui pra frente não iremos empacotar o conjunto de caracteres e a collation no mesmo objeto.

Existe um tratamento especial do conjunto de caracteres nacional no MySQL 4.1. `NCHAR` não é o mesmo que `CHAR` e literais `N'...'` não são o mesmo dos literais `'...'`.

Finalmente, existe um formato de arquivo diferente para armazenar informações sobre conjunto de caracteres e collation. Esteja certo que você reinstalou o diretório '`/share/mysql/charsets/`' contendo o novo arquivo de configurações.

Se você quiser iniciar o `mysqld` de uma distribuição 4.1.x com dados craidos pelo MySQL 4.0, você deve iniciar o servidor com o mesmo conjunto de caracteres e collation. Neste caso você não precisará de reindexar os dados.

Existem dois modos de fazê-lo:

```
shell> ./configure --with-character-set=... --with-collation=...
shell> ./mysqld --default-character-set=... --default-collation=...
```

Se você usou o `mysql` com, por exemplo, oconjunto de caracteres `danish` do MySQL 4.0, você agora deve usar o conjunto de caracteres `latin1` e a collation `latin1_danish_ci`:

```
shell> ./configure --with-character-set=latin1
           --with-collation=latin1_danish_ci
shell> ./mysqld --default-character-set=latin1
           --default-collation=latin1_danish_ci
```

Use a tabela mostrada na próxima seção para encontrar o nome do antigo conjunto de caracteres do MySQL 4.0 e o par conjunto de caracteres/collation equivalente no MySQL 4.1.

9.10.1 Conjunto de Caracteres do MySQL e o Par/Conjunto de Caracter/Collation Correspondente do MySQL 4.1

ID	Conjunto de Caracter - 4.0	Conjunto de Caracter - 4.1	Collation - 4.1
1	big5	big5	big5_chinese_ci
2	czech	latin2	latin2_czech_ci
3	dec8	dec8	dec8_swedish_ci
4	dos	cp850	cp850_general_ci
5	german1	latin1	latin1_german1_ci
6	hp8	hp8	hp8_english_ci
7	koi8_ru	koi8r	koi8r_general_ci
8	latin1	latin1	latin1_swedish_ci
9	latin2	latin2	latin2_general_ci
10	swe7	swe7	swe7_swedish_ci
11	usa7	ascii	ascii_general_ci
12	ujis	ujis	ujis_japanese_ci
13	sjis	sjis	sjis_japanese_ci
14	cp1251	cp1251	cp1251_bulgarian_ci
15	danish	latin1	latin1_danish_ci
16	hebrew	hebrew	hebrew_general_ci
17	win1251	(removed)	(removed)
18	tis620	tis620	tis620_thai_ci
19	euc_kr	euckr	euckr_korean_ci
20	estonia	latin7	latin7_estonian_ci
21	hungarian	latin2	latin2_hungarian_ci
22	koi8_ukr	koi8u	koi8u_ukrainian_ci
23	win1251ukr	cp1251	cp1251_ukrainian_ci
24	gb2312	gb2312	gb2312_chinese_ci
25	greek	greek	greek_general_ci
26	win1250	cp1250	cp1250_general_ci
27	croat	latin2	latin2_croatian_ci
28	gbk	gbk	gbk_chinese_ci
29	cp1257	cp1257	cp1257_lithuanian_ci
30	latin5	latin5	latin5_turkish_ci
31	latin1_de	latin1	latin1_german2_ci

9.11 Os conjuntos de Caracteres e Collations que o MySQL Suporta

Aqui está uma lista do conjunto de caracter e collation que o MySQL suporta. Como as opções e configuração de instalação diferem, alguns sites não terão todos os itens da lista, e alguns sites terão itens que não estão na lista porque a definição de novos conjunto de caracteres e collation é direto.

O MySQL suporta mais de 70 collations e mais de 30 conjunto de caracteres.

```
mysql> SHOW CHARACTER SET;
+----------+-----------------------------+---------------------+--------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+--------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |      2 |
| dec8     | DEC West European           | dec8_swedish_ci     |      1 |
| cp850    | DOS West European           | cp850_general_ci    |      1 |
| hp8      | HP West European            | hp8_english_ci      |      1 |
| koi8r    | KOI8-R Relcom Russian       | koi8r_general_ci    |      1 |
| latin1   | ISO 8859-1 West European    | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |      1 |
| swe7     | 7bit Swedish                | swe7_swedish_ci     |      1 |
| ascii    | US ASCII                    | ascii_general_ci    |      1 |
| ujis     | EUC-JP Japanese             | ujis_japanese_ci    |      3 |
| sjis     | Shift-JIS Japanese          | sjis_japanese_ci    |      2 |
| cp1251   | Windows Cyrillic            | cp1251_bulgarian_ci |      1 |
| hebrew   | ISO 8859-8 Hebrew           | hebrew_general_ci   |      1 |
| tis620   | TIS620 Thai                 | tis620_thai_ci      |      1 |
| euckr    | EUC-KR Korean               | euckr_korean_ci     |      2 |
| koi8u    | KOI8-U Ukrainian            | koi8u_general_ci    |      1 |
| gb2312   | GB2312 Simplified Chinese   | gb2312_chinese_ci   |      2 |
| greek    | ISO 8859-7 Greek            | greek_general_ci    |      1 |
| cp1250   | Windows Central European    | cp1250_general_ci   |      1 |
| gbk      | GBK Simplified Chinese      | gbk_chinese_ci      |      2 |
| latin5   | ISO 8859-9 Turkish          | latin5_turkish_ci   |      1 |
| armscii8 | ARMSCII-8 Armenian          | armscii8_general_ci |      1 |
| utf8     | UTF-8 Unicode               | utf8_general_ci     |      3 |
| ucs2     | UCS-2 Unicode               | ucs2_general_ci     |      2 |
| cp866    | DOS Russian                 | cp866_general_ci    |      1 |
| keybcs2  | DOS Kamenicky Czech-Slovak  | keybcs2_general_ci  |      1 |
| macce    | Mac Central European        | macce_general_ci    |      1 |
| macroman | Mac West European           | macroman_general_ci |      1 |
| cp852    | DOS Central European        | cp852_general_ci    |      1 |
| latin7   | ISO 8859-13 Baltic          | latin7_general_ci   |      1 |
| cp1256   | Windows Arabic              | cp1256_general_ci   |      1 |
| cp1257   | Windows Baltic              | cp1257_general_ci   |      1 |
| binary   | Binary pseudo charset       | binary              |      1 |
+----------+-----------------------------+---------------------+--------+
33 rows in set (0.01 sec)
```

NB: TODOS OS CONJUNTO DE CARACTERES TEM UMA COLLATION BINÁRIA. NÃO INCLUÍMOS A COLLATION BINÁRIA EM TODAS AS DESCRIÇÕES A SEGUIR.

9.11.1 O Conjunto de Caracteres Unicode

É claro que existem os nossos dois conjuntos de caracteres Unicode. Você pode armazenar texto em cerca de 650 línguas usando estes conjunto de caracteres. Não adicionamos um grande número de collations para estes dois novos conjuntos ainda, mas isto acontecerá logo. Agora eles possuem a collation caso-insensitivo e acento-insensitivo, mais a collation binária.

Charset	Description	Default collation	Maxlen
utf8	UTF-8 Unicode	utf8_general_ci	3
ucs2	UCS-2 Unicode	ucs2_general_ci	2

9.11.2 Conjunto de Caracteres para Plataformas Específicas

Charset	Description	Default collation	Maxlen
dec8	DEC West European	dec8_swedish_ci	1
hp8	HP West European	hp8_english_ci	1

9.11.3 Conjunto de Caracteres do Sul da Europa e Oriente Médio

Charset	Description	Default collation	Maxlen
armscii8	ARMSCII-8 Armenian	armscii8_general_ci	1
cp1256	Windows Arabic	cp1256_general_ci	1
hebrew	ISO 8859-8 Hebrew	hebrew_general_ci	1
greek	ISO 8859-7 Greek	greek_general_ci	1
latin5	ISO 8859-9 Turkish	latin5_turkish_ci	1
geostd8	Georgian	geostd8_general_ci	1

9.11.4 Os Conjuntos de Caracteres Asiáticos

O conjunto de caracteres Asiático que suportamos inclui Chinês, Japonês, Coreano e Tailandês. Estes podem ser complicados. Por exemplo, o conjunto Chinês devem permitir milhares de caracteres diferentes.

Charset	Description	Default collation	Maxlen

```
+----------+-----------------------------+--------------------+--------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci    |      2 |
| gb2312   | GB2312 Simplified Chinese   | gb2312_chinese_ci  |      2 |
| gbk      | GBK Simplified Chinese      | gbk_chinese_ci     |      2 |
| euckr    | EUC-KR Korean               | euckr_korean_ci    |      2 |
| ujis     | EUC-JP Japanese             | ujis_japanese_ci   |      3 |
| sjis     | Shift-JIS Japanese          | sjis_japanese_ci   |      2 |
| tis620   | TIS620 Thai                 | tis620_thai_ci     |      1 |
+----------+-----------------------------+--------------------+--------+
```

9.11.5 Os Conjuntos de Caracteres Bálticos

O conjunto de caracter Báltico cobre as linguagens da Estonia, Letônia e Lituânia. Existem dois conjunto de caracteres Bálticos suportados:

- `latin7` (ISO 8859-13 Baltic):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| latin7_estonian_cs   | latin7   | 20 |         |          |       0 |
| latin7_general_ci    | latin7   | 41 | Yes     |          |       0 |
| latin7_general_cs    | latin7   | 42 |         |          |       0 |
| latin7_bin           | latin7   | 79 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

- `cp1257` (Windows Baltic):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| cp1257_lithuanian_ci | cp1257   | 29 |         |          |       0 |
| cp1257_bin           | cp1257   | 58 |         |          |       0 |
| cp1257_general_ci    | cp1257   | 59 | Yes     |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

9.11.6 Os Conjuntos de Caracteres Cirílicos

Aqui estão os conjunto de caracteres e collation cirílicos para uso com as linguagens Belarússia, Búlgaro, Russo e Ucraniano.

- `cp1251` (Windows Cyrillic):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| cp1251_bulgarian_ci  | cp1251   | 14 |         |          |       0 |
| cp1251_ukrainian_ci  | cp1251   | 23 |         |          |       0 |
| cp1251_bin           | cp1251   | 50 |         |          |       0 |
| cp1251_general_ci    | cp1251   | 51 | Yes     |          |       0 |
| cp1251_general_cs    | cp1251   | 52 |         |          |       0 |
```

```
+----------------------+----------+----+---------+----------+---------+
```

- `cp866` (DOS Russian):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| cp866_general_ci     | cp866    | 36 | Yes     |          |       0 |
| cp866_bin            | cp866    | 68 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

- `koi8r` (KOI8-R Relcom Russian, primarily used in Russia on Unix):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| koi8r_general_ci     | koi8r    |  7 | Yes     |          |       0 |
| koi8r_bin            | koi8r    | 74 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

- `koi8u` (KOI8-U Ukrainian, primarily used in Ukraine on Unix):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| koi8u_general_ci     | koi8u    | 22 | Yes     |          |       0 |
| koi8u_bin            | koi8u    | 75 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

9.11.7 O Conjunto de Caracteres da Europa Central

Temos algum suporte para conjunto de caracteres usados na República Tcheca, Eslováquia, Hungria, Romênia, Eslovênia, Croácia e Polônia.

- `cp1250` (Windows Central European):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| cp1250_general_ci    | cp1250   | 26 | Yes     |          |       0 |
| cp1250_czech_ci      | cp1250   | 34 |         | Yes      |       2 |
| cp1250_bin           | cp1250   | 66 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

- `cp852` (DOS Central European):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| cp852_general_ci     | cp852    | 40 | Yes     |          |       0 |
| cp852_bin            | cp852    | 81 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

- `macce` (Mac Central European):

```
+----------------------+----------+----+---------+----------+---------+
```

```
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| macce_general_ci     | macce    | 38 | Yes     |          |       0 |
| macce_bin            | macce    | 43 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

- `latin2` (ISO 8859-2 Central European):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| latin2_czech_ci      | latin2   |  2 |         | Yes      |       4 |
| latin2_general_ci    | latin2   |  9 | Yes     |          |       0 |
| latin2_hungarian_ci  | latin2   | 21 |         |          |       0 |
| latin2_croatian_ci   | latin2   | 27 |         |          |       0 |
| latin2_bin           | latin2   | 77 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

- `keybcs2` (DOS Kamenicky Czech-Slovak):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| keybcs2_general_ci   | keybcs2  | 37 | Yes     |          |       0 |
| keybcs2_bin          | keybcs2  | 73 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

9.11.8 Os Conjuntos de Caracteres da Europa Ocidental

O Cojunto de Caracteres da Europa Ocidental cobre a maioria das linguagens desta região como Francês, Espanhol, Catalão, Basco, Português, Italiano, Albanês, Holandês, Alemão, Finlandes, Dinamarquês, Sueco, Norueguês, Faroese, Islandês, Irlandês, Escocês e Inglês

- `latin1` (ISO 8859-1 West European):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| latin1_german1_ci    | latin1   |  5 |         |          |       0 |
| latin1_swedish_ci    | latin1   |  8 | Yes     | Yes      |       0 |
| latin1_danish_ci     | latin1   | 15 |         |          |       0 |
| latin1_german2_ci    | latin1   | 31 |         | Yes      |       2 |
| latin1_bin           | latin1   | 47 |         | Yes      |       0 |
| latin1_general_ci    | latin1   | 48 |         |          |       0 |
| latin1_general_cs    | latin1   | 49 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

  A collation `latin1_swedish_ci` é o padrão que provavelmente é usado pela maioria dos usuários do MySQL. É constantemente indicado que ele é baseado nas regras de collation do Suéco/Finlandês, mas você encontrará Suécos e Finlandeses que descordam desta afirmação.

As collations `latin1_german1_ci` e `latin1_german2_ci` são baseadas nos padrões DIN-1 e DIN-2, onde DIN significa Deutsches Institut für Normung (isto é, a resposta Alemã ao ANSI). DIN-1 é chamada collation de dicionário e o DIN-2 é chamado a collation de agenda.

- Regras `latin1_german1_ci` (dicionários):

  'Ä' = 'A', 'Ö' = 'O', 'Ü' = 'U', 'ß' = 's'

- Regras `latin1_german2_ci` (agendas):

  'Ä' = 'AE', 'Ö' = 'OE', 'Ü' = 'UE', 'ß' = 'ss'

- `macroman` (Mac West European):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| macroman_general_ci  | macroman | 39 | Yes     |          |       0 |
| macroman_bin         | macroman | 53 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

- `cp850` (DOS West European):

```
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| cp850_general_ci     | cp850    |  4 | Yes     |          |       0 |
| cp850_bin            | cp850    | 80 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
```

10 Extensões Espacias em MySQL

O MySQL 4.1 introduz extensões espaciais para permitir gerar, armazenar e analisar recursos geográficos. Atualmente estes recursos estão disponiveis apenas para tabelas MyISAM. Este capítulo cobre os seguintes tópicos:

- A base destas extensões espaciais no modelo OpenGIS
- Formato de dados para representação de dados espaciais
- Como usar dados espaciais no MySQL
- Uso do índice para dados espaciais
- Diferenças do MySQL para a especificação OpenGIS

10.1 Introdução

O MySQL implementea extensões espaciais seguindo especificações do `Open GIS Consortium` (OGC). Este é um consórcio internacional com mais de 250 companhias, agências, universidades participando no desenvolvimento de soluções conceituais disponíveis publicamente que podem der úteis com todos os tipos de aplicações que gerenciam dados espaciais. O OGC mantém um web site em `http://www.opengis.org/`.

Em 1997, o Open GIS Consortium publicou o *OpenGIS ® Simple Features Specifications For SQL* (Especificações de Recursos OpenGIS ® Simples Para SQL), um documento que propos diversos modos conceituais de para extender um SQL RDBMS para suportar dados espaciais. Esta especificação está disponível no web site do OpenGIS em `http://www.opengis.org/techno/implementation.htm`. Ele contém informações adicionais relevantes a este capítulo.

O MySQL implementa um subconjunto do ambiente **SQL com Tipos Geométricos** proposto pela OGC. Este termo se refere a um ambiente SQL que tem sido extendido com um conjunto de tipos geomértricos. Uma coluna SQL com valor geométrico é implementada como uma coluna de um tipo geométrico. As especificações descrevem um conjunto de tipod geométricos do SQL, bem como funções deste tipo para criar e analisar valores geométricos.

Um **recurso geográfico** é qualquer coisa no mundo que tem uma posição.

Um recurso pode ser:

- Uma entidade. Por exemplo, uma montanha, uma lagoa, em cidade
- Um espaço. Por exemplo, um área de código postal, os trópicos
- Uma localização definida. Por exemplo, um cruzamento. como um lugar específico onde duas ruas se interceptam.

Você também pode encontrar documentos que utilizam o termo **recurso geoespacial** para se referir a recursos geográficos.

Geometria é outra palavra que denota um recurso geográfico. O significado original da palavra **geometria** denota um ramo da matemática. Outro significado vindo da cartografia, se referem aos recursos geométricos que os cartógrafos usam para mapear o mundo.

Este capítulo utiliza todos estes termos como sinônimo: **recurso geográfico**, **recurso geoespacial**, **recurso** ou **geometria**, O termo normalmente mais usado aqui é **geometry**.

Vamos definir uma **geometria** como *um ponto ou um agregado de pontos representando alguma coisa no mundo que possui uma localização.*

10.2 O Modelo Geomátrico OpenGIS

O conjunto de tipos geométricos, proposto pelo ambiente **SQL com Tipos Geométricos** da OGC, é base do **Modelo Geométrico OpenGIS**. Neste modelo, cada objeto geométrico tem as seguintes propriedades gerais:

- é associado com um Sistema de Referência Espacial, que descreve a coordenada espacial, na qual o objeto é definido.
- pertence a alguma classe geométrica.

10.2.1 A Hierarquia da Classe `Geometry`

As classes geometry definem uma hierarquia como a seguir:

- `Geometry` (não-instanciável)
  - `Point` (instanciável)
  - `Curve` (não-instanciável)
    - `LineString` (instanciável)
      - `Line`
      - `LinearRing`
  - `Surface` (não-instanciável)
    - `Polygon` (instanciável)
  - `GeometryCollection` (instanciável)
    - `MultiPoint` (instanciável)
    - `MultiCurve` (não-instanciável)
      - `MultiLineString` (instanciável)
    - `MultiSurface` (não-instanciável)
      - `MultiPolygon` (instanciável)

Algumas destas classes são abstratas (não-instanciável). Isto é, não é possível criar um objeto desta classe. Outras classes são instanciáveis e objetos podem ser criados deles. Cada classe tem propriedades e podem ter declarações (regras que definem intâncias de classes válidas).

`Geometry` é a classe base. É uma classe abstrata (não-instanciável). As subclasses instanciáveis de `Geometry` são restritas a objetos geométricos de zero, uma e duas dimensões que existem no espeaço de coordenadas bidimensional. Todas as classes geométricas instanciáveis são definidas para que instâncias válidas da classe geometry são topologicamente fechados (isto é, todas as geometrias definidas incluem seus limites).

A classe base `Geometry` tem subclasses para `Point`, `Curve`, `Surface` e `GeometryCollection`:

- `Point` representam objetos sem dimensão.
- `Curve` representam para objetos de uma dimensão, e tem a subclasse `LineString`, com subclasses `Line` e `LinearRing`.
- `Surface` é criado para objetos bidimensionais e tem a subclasse `Polygon`.

- `GeometryCollection` tem classes de coleção com zero-, uma- e duas-dimensões chamadas `MultiPoint`, `MultiLineString` e `MultiPolygon` para modelagem geométrica correspondente a coleções de `Points`, `LineStrings` e `Polygons` respectivamente. `MultiCurve` e `MultiSurface` são introduzidas como superclasses abastratas que generalizam a interface de coleção para tratar `Curves` e `Surfaces`.

`Geometry`, `Curve`, `Surface`, `MultiCurve` e `MultiSurface` são definidos como classes não instanciáveis. Eles definem em conjunto de métodos comuns para suas subclasses e incluidos por razões de extensabilidade.

`Point`, `LineString`, `Polygon`, `GeometryCollection`, `MultiPoint`, `MultiLineString`, `MultiPolygon` são classses instanciáveis.

10.2.2 Classe `Geometry`

`Geometry` é a classe raiz da hierarquia. É uma classe não instanciável mas possui vária propriedades comuns a todos os valores de geometria de qualquer das subclasses `Geometry`. Estas propriedades estão descritas na lista a seguir (Subclasses particulares tem as suas próprias propriedades específicas, descritas posteriormente):

Propriedades de geometria

Um valor geometry tem as seguintes propriedades:

- É o tipo (**type**). Cada geometria pertence a uma das classes instanciáveis na hierarquia.
- Seu **SRID** ou Identificador de Referência Espacial. Este valor identifica o Sistema de Referência Espacial associada da geometria, o qual descreve o coordenada espacial na qual objeto geomtrico está definido.
- Coordenadas (**coordinates**) em seu Sistema de Referência Espacial, representado por um número de precisão dupla (8 byte). Todas as geometrias não-vazias incluem pelo menos um par de coordenadas (X,Y). Geometrias vazias não contem cooredenadas.

  Coordenadas estão relacionadas ao SRID. Por exemplo, em sistemas de coordenadas diferentes, a distância entre dois objetos podem diferir mesmo quando os objetos têm as mesmas coordenadas, porque as distâncias no sistema de coordenadas **planar** e a distância no sistema **geocentrico** (coordenadas na superfície da Terra) são coisas diferentes.
- Seu interior (**interior**), limite (**boundary**) e exterior (**exterior**).

  Todas as geometrias ocupam alguma porção no espaço. O exterior de uma geometria é todo espaço não ocupado pela geometria. O interiro é o espeaço ocupado pela geometria. O limite é a interface entre o interior e o exterior
- Seu **MBR** (Retângulo de Limite Mínimo - Minimum Bounding Rectangle), ou Envelope, da geometria. Este é a geometria limitar, formado pelas coordenadas de mínimo e máximo (X,Y):

  ```
  ((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
  ```
- A qualidade de ser **simple** ou **non-simple** (simples ou não simples). Valores geometricos alguns tipos (`LineString`, `Multipoint`, `MultiLineString`) podem ser simples ou não-simples. Cada tipo determina sua prórpia afirmação de ser simples ou não-simples.

- A qualidade de ser **closed** ou **not closed** (fechado ou não fechado). Valores geométricos de alguns tipos (`LineString`, `MultiString`) podem ser fechado ou não fechado. Cada tipo determina a sua própria afirmação de ser fechado ou não fachado.
- A qualidade de ser **empty** ou **not empty** (vazio ou não vazio). Uma geometria é vazia se ela não tem nenhum ponto. Exterior, interior e limite de ma geometria vazia não estão definidos. (isto é, eles são representados por valores **NULL**). Uma geometria vazia é definida sempre simples e ter um área de 0.
- Sua dimensão (**dimension**). Uma geometria pode ter uma dimensão de −1, 0, 1 or 2:
  - −1 usado para geometrias vazias
  - 0 usado para geometrias sem tamanho e sem area.
  - 1 usado para geometrias com tamanho diferente de zero e sem area.
  - 2 usado para geometrias com area diferente de zero.

  `Points` tem uma dimensião de zero. `LineStrings` tem uma dimensão de 1. `Polygons` tem uma dimensão de 2. Dimensões de `MultiPoints`, `MultiLineStrings` e `MultiPolygons` são a é mesma da dimensão dos elementos dos quais eles consistem.

10.2.3 Classe `Point`

Um `Point` é uma geometria que representa um único local no espaço coordenado.

Exemplos de `Point`

- Imagine um mapa do munod de larga-escala com muitas cidades. Um ponto poderia representar cada cidade.
- Em um mapa da cidade, um `Point` poderia epresntar uma parada de onibus.

Propriedades de `Point`

- Valor de coordenada X.
- Valor da coordenada Y.
- O `Point` é definido como uma geometria de dimensão zero.
- O limite de um `Point` é um conjunto vazio.

10.2.4 Classe `Curve`

Uma `Curve` é uma geometria unidimensional, normalmente representado por uma sequência de pontos. Subclasses particulares de `Curve` define o tipo de interpolação entre pontos. `Curve` é uma classe não-instanciável.

Propriedades de `Curve`

- As coordenadas de seus pontos.
- `Curve` é definiido como uma geometria unidimensional.

- A `Curve` é simples (simple) se ela não passa pelo mesmo ponto duas vezes.
- A `Curve` é fechada (closed) se o ponto inicial é igual ao ponto final.
- O limite (boundary) de uma `Curve` fechada é vazio.
- O limite (boundary) de uma `Curve` não-fachada cociste do seus dois pontos finais.
- A `Curve` que é simples (simple) e fechada (closed) é uma `LinearRing`.

10.2.5 Classe `LineString`

Uma `LineString` é uma `Curve` com interpolação linear entre pontos.

Exemplos de `LineString`

- Em um mapa mundi uma `LineStrings` poderia representar os rios.
- Um um mapa da cidade uma `LineStrings` poderia respresntar ruas.

Propriedades `LineString`

- Coordenadas de segmentos `LineString` definidos por cada par de pontos consecutivos.
- Uma `LineString` é uma `Line`, se ela consiste de exatamente dois pontos.
- A `LineString` é uma `LinearRing`, se for fechada (closed) e simples (simple).

10.2.6 Classe `Surface`

Uma `Surface` é uma geometria bidimensional. Ele é uma classe não instanciável. Sua única subclasse instanciável é `Polygon`.

Propriedades de `Surface`

- Uma `Surface` é definida com uma geomtria bidimensional.
- A especificação OpenGIS define uma `Surface` simples como uma geometria que consiste de um único 'patch' que é associado com um 'exterior boundary' (limite exterior) e zero ou mais 'interior' boundaries (limites interiores).
- O limite (boundary) de uma `Surface` simples é o conjunto de curvas fechadas correspondente a seus limites exterior e interior.

10.2.7 Classe `Polygon`

Um `Polygon` é uma `Surface` planar representando uma geometria multi-lados. Ela é definida por um limite exterior e zero ou mais limites interiores, onde cada limite interior define um buraco no `Polygon`.

Exemplos de `Polygon`

- Em um mapa de região, objetos `Polygon` podem representar florestas, distritos, etc.

As afirmações para os polygons (as regras que definem polygons válidos) são:

1. O limite (boundary) de um `Polygon` consiste de um conjunto de LinearRings (ex. `LineString`s que são simples e fechadas) que fazem os seus limites interior e exterior.
2. Dois aneis no limite não podem se cruzar. Os aneis no limite de um `Polygon` podem se interseptar em um `Point`, mas apenas como uma tangente.
3. Um `Polygon` não pode ter linhas cortadas, pontas ou cavidades.
4. O interior de cada `Polygon` e um conjunto de pontos conectados.
5. O Exterior de um `Polygon` com um ou mais buracos não está conectado. Cada buraco define um componenete conectados do exterior.

Nas afirmações acimas, poligonos são geometrias simples. Estas afirmações fazem de um `Polygon` uma geometria simples.

10.2.8 Classe `GeometryCollection`

Um `GeometryCollection` é uma geometria que é um coleção de um ou mais geometrias de qualquer classe.

Todos os elementos em uma `GeometryCollection` deve estar no mesmo Sistema de Referência Espacial (ex. no mesmo sistema de coordenadas). GeometryCollection não coloca nenhuma outra restrição em seus elementos, embora as subclasses de `GeometryCollection` descritas abaixo possam restringir membros com base em:

- Tipo de Elementos (por exemplo, um `MultiPoint` pode conter apenas elementos `Point`
- Dimensão.
- Restrições no grau de sobreposição espacial entre elementos.

10.2.9 Classe `MultiPoint`

Um `MultiPoint` é uma coleção de geometrias compostas de elementos `Point`. Os pontos não estão conectados ou ordenados de forma alguma.

Exemplos de `MultiPoint`

- Em um mapa mundi, um Multipoint podia representar uma cadeia de pequenas ilhas.

Propriedades de `MultiPoint`

- `MultiPoint` é definido com uma geometria sem dimensão.
- Um `MultiPoint` é simples se não há dois valores de seus `Point` iguais no `MultiPoint` (tem valores de coordenadas iguais).
- O limite (boundary) de um `MultiPoint` é um conjunto vazio.

10.2.10 Classe `MultiCurve`

Uma `MultiCurve` é uma coleção de geometria compostas de elementos `Curve`. `MultiCurve` é uma classe não instanciável.

Propriedades de `MultiCurve`

- A `MultiCurve` é definida como uma geometria de uma dimensão.
- A `MultiCurve` é simples se e somente se todos os seus elementos são simples, a única interseção entre quaisquer dois elementos ocorrem entre pontos que estão nos limites (boundaries) de ambos os elementos.
- O limite (boundary) de uma `MultiCurve` é obtida aplicando a "mod 2 union rule": Um ponto está no limite (boundary) de uma `MultiCurve` se ele está no limite de um número ímpar de elementos da `MultiCurve`.
- Um `MultiCurve` é fechado se todos os seus elementos são fechados.
- O limite de uma `MultiCurve` fechada e sempre vazio.

10.2.11 Classe `MultiLineString` (Multi Linhas)

Um `MultiLineString` é uma coleção de geométrias `MultiCurve` composto de elementos `LineString`.

`MultiLineString`

- Em uma mapa regional, um `MultiLineString` pode represntar um rede hidrografica ou uma malha de rodovias.

10.2.12 Classe `MultiSurface` (Multi Superfícies)

Um `MultiSurface` é uma coleção geometrica compostos de elementos de superfície MultiSurface é uma classe não instanciável. Sua única subclasse instanciável é `MultiPolygon`

Afirmações de `MultiSurface`

1. O interior de quaisquer duas superfícies em uma `MultiSurface` não podem se interceptar.
2. O limite de quaiqsquer dois elementos em um `MultiSurface` podem interceptar em um número finito de pontos.

10.2.13 Classe `MultiPolygon` (Multi Polígonos)

Um `MultiPolygon` é um objeto `MultiSurface` compostos de elementos `Polygon`.

Exemplos de `MultiPolygon`

- Em um mapa regional, um `MultiPolygon` pode representar um sistema de lagos.

As afirações dos `MultiPolygons` são:

1. O interior de dois valores `Polygon` que são elementos de um `MultiPolygon` não podem interceptar.
2. Os limites (Boundaries) de quaisquer dois valores `Polygon` que são elementos de um `MultiPolygon` não podem cruzar e pode se tocar em um número finito de pontos. (O cruzamento também é proibido pela primeira afirmação.)
3. Um `MultiPolygon` não pode ter linhas cortadas, pontas ou cavidades. Um MultiPolygon é um conjunto de pontos regular e fechado.
4. O interior de um `MultiPolygon` composto por mais de um `Polygon` não está conectado, o número de componentes conectados do interior de um `MultiPolygon` é igual ao número de valores `Polygon` no `MultiPolygon`.

Propriedades de `MultiPolygon`

- MultiPolygon é definido como uma geometria bidimensional.
- O limite (boundary) de um `MultiPolygon` é um conjunto de curvas fechadas (valores `LineStrings`) correspondente ao limite dos valores seus elementos `Polygon`.
- Cada `Curve` no limite do `MultiPolygon` este no limite de exatamente um elemento `Polygon`.
- Toda `Curve` no limite de um elemento Polygon está no limite do `MultiPolygon`.

10.3 Formatos de Dados Espaciais Suportados

Esta seção descreve o formato de dados espaciais padrão que são utilizados para representar objetos geometry em consultas.

Eles são:

- Formato Well-Known Text (WKT).
- Formato Well-Known Binary (WKB).

Internamente, o MySQL armazena valores geometry em um formato que não é identico nem ao format WKT ou WKB.

10.3.1 Formato Well-Known Text (WKT)

A representação Well-Known Text (WKT) de `Geometry` é criada para troca de dados de geometria na forma ASCII.

Exemplos de representações WKT representations de objetos geometry são:

- Um `Point` (ponto).

  ```
  POINT(15 20)
  ```

  Note que pontos coordenados são especificados sem separção por vírgulas.
- Um `LineString` (linha) com quatro pontos.

  ```
  LINESTRING(0 0, 10 10, 20 25, 50 60)
  ```

- Um `Polygon` (polígono) com um anel exterior e um anél interior.

  ```
  POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))
  ```

- Um `MultiPoint` (multipontos) com três valores `Points`.

  ```
  MULTIPOINT(0 0, 20 20, 60 60)
  ```

- Um `MultiLineString` (multi linhas) com dois valores `LineString`.

  ```
  MULTILINESTRING((10 10, 20 20), (15 15, 30 15))
  ```

- Um `MultiPolygon` (multi polígonos) com dois valores `Polygon`.

  ```
  MULTIPOLYGON(((0 0,10 0,10 10,0 10,0 0)),((5 5,7 5,7 7,5 7, 5 5)))
  ```

- Um `GeometryCollection` (Coleção de Geometria) consistindo de dois valores `Points` e um `LineString`.

  ```
  GEOMETRYCOLLECTION(POINT(10 10), POINT(30 30), LINESTRING(15 15, 20 20))
  ```

Uma gramática Backus-Naur que especifica as regras de produção formal para gravar valores WKT podem ser encontrados na documentação de especificação OGC indicada próximo ao início deste capítulo.

10.3.2 Formato Well-Known Binary (WKB)

A representação Well-Known Binary (WKB) para valores geométricos é definida pela especificação OpenGIS. Ela também é definida no padrão ISO "SQL/MM Part 3: Spatial".

WKB é usado para trocar dados geometry como fluxos binários representados por valores BLOB contendop informações geométricas WKB.

WKB usa inteiros sem sinal de 1-byte e 4-byte e números de precisão dupla de 8-byte (formato IEEE 754). Um byte é 8 bits.

Por exemplo, um valor WKB que corresonde a `POINT(1 1)` consiste desta sequência de 21 bytes (cada um representado aqui por dois digitos hexa):

```
010100000000000000000F03F000000000000F03F
```

A sequência pode ser quebrada nestes componentes:

```
Byte order : 01
WKB type   : 01000000
X          : 000000000000F03F
Y          : 000000000000F03F
```

A respresentação do componente está a seguir:

- O byte order pode ser de 0 ou 1 para indicar o tipo little-endian ou big-endian. Os byte orders little-endian e big-endian também são conhecidos como Network Data Representation - Representação de Dados de Rede (NDR) e External Data Representation - Representação de Dados Externos (XDR), repectivamente.
- O tipo WKB é um código que indica o tipo de geometria. Valores de 1 a 7 indicam `Point`, `LineString`, `Polygon`, `MultiPoint`, `MultiLineString`, `MultiPolygon`, e `GeometryCollection`.
- Um valor `Point` têm coordenadas X e Y, cada uma representada como um valor de dupla precisão.

Valores WKB para valores de geometria mais complexas são representados por estrutras de dados mais complexas, como detalhado na epecificação OpenGIS.

10.4 Criando um Banco de Dados MySQL Habilitado Espacialmente

Esta seção descreve os tipos de dados que você pode usar para representar dados espaciais no MySQL e as funções disponíveis para criar e recuperar valores espaciais.

10.4.1 Tipos de Dados Espaciais do MySQL

MySQL fornece um hierarquia de tipos de dados que correspondem as classes na hierarquia de classes do Modelo Geometrico OpenGIS. Alguns destes tipos guardam valores de geometria únicos:

- `GEOMETRY`
- `POINT`
- `LINESTRING`
- `POLYGON`

O tipo `GEOMETRY` é o mais genérico destes tipos, ele pode armazenar geometrias de qualquer tipo. Os outros tipos restringem seus valores a tipos de geometria específicos.

Os outros tipos de dados tem coleções de valores:

- `MULTIPOINT`
- `MULTILINESTRING`
- `MULTIPOLYGON`
- `GEOMETRYCOLLECTION`

`GEOMETRYCOLLECTION` pode armazenar uma coleçao de objetos de qualquer tipo. Os outros tipos de coleções restrigem o tipo dos membros da coleção para um tipo de geometria específico.

10.4.2 Criando Valores Espaciais

Esta seção descreve como criar valores espaciais usando as funções Well-Known Text e Well-Known Binary que estão definidas no padrão OpenGIS, e usando funções específicas do MySQL.

10.4.2.1 Criando Valores Geometry Usando Funções WKT

O MySQL fornece algumas funções que utilizam a representação Well-Known Text (e, opcionalmente, um identificador sistema de referência espacial (SRID)) e retorna a geometria correspondente.

`GeomFromText()` aceita um WKT de qualquer tipo de geometria com seu primeiro argumento. Uma implementação também fornece uma função de construção específica do tipo para cada tipo de geometria.

`GeomFromText(wkt[,srid])`
`GeometryFromText(wkt[,srid])`

Controi um valor geometria de qualquer tipo usando sua representação WKT e SRID.

`PointFromText(wkt[,srid])`
: Controi um valor `POINT` usando sua representação WKT e SRID.

`LineFromText(wkt[,srid])`
`LineStringFromText(wkt[,srid])`
: Constroi um valor `LINESTRING` usando sua representação WKT e SRID.

`PolyFromText(wkt[,srid])`
`PolygonFromText(wkt[,srid])`
: Constroi um valor `POLYGON` usasdo sua representação WKT e SRID.

`MPointFromText(wkt[,srid])`
`MultiPointFromText(wkt[,srid])`
: Contrói um valor `MULTIPOINT` usando sua representação WKT e SRID.

`MLineFromText(wkt[,srid])`
`MultiLineStringFromText(wkt[,srid])`
: Contrói um valor `MULTILINESTRING` usando sua representação WKT e SRID.

`MPolyFromText(wkt[,srid])`
`MultiPolygonFromText(wkt[,srid])`
: Contrói um valor `MULTIPOLYGON` usando sua representação WKT e SRID.

`GeomCollFromText(wkt[,srid])`
`GeometryCollectionFromText(wkt[,srid])`
: Constrói um valor `GEOMETRYCOLLECTION` usando sua representação WKT e SRID.

A especificação OpenGIS também descreve funções opcionais para construção de valores `Polygon` ou `MultiPolygon` baseados na representação WKT de uma coleção de anéis ou valores `LineString` fechados. Estes valores podem se interceptar. OMySQL ainda não implementou estas funções:

`BdPolyFromText(wkt,srid)`
: Constrói um valor `Polygon` a partir de um valor `MultiLineString` no formato WKT contendo uma coleção arbitrária de valores `LineString` fechados.

`BdMPolyFromText(wkt,srid)`
: Constrói um valor `MultiPolygon` a partir de um valor `MultiLineString` no formato WKT contendo uma coleção arbitrária de vlaores `LineString` fechados.

10.4.2.2 Criando Valores Geometry Usando Funções WKB

O MySQL fornece um conjunto de funções que utilizam um BLOB contendo representação Well-Known Binary (e, opcionalmente, um indentificador de sistema de referência espacial (SRID)), e retornam a geometria correspondente.

`GeomFromWKT` pode acitar um WKB de qualquer tipo de geometria como seu primeiro argumento. Uma implementação também fornece uma função de construção específica para cada tipo de geometria como descrito na lista acima.

`GeomFromWKB(wkb,srid)`
`GeometryFromWKB(wkt,srid)`
: Constrói um valor geometria de qualquer tipo usando seua representação WKB e SRID.

`PointFromWKB(wkb[,srid])`
: Constrói um valor `POINT` usando sua representação WKB e SRID.

`LineFromWKB(wkb[,srid])`
`LineStringFromWKB(wkb[,srid])`
: Constrói um valor `LINESTRING` usando sua representação WKB e SRID.

`PolyFromWKB(wkb[,srid])`
`PolygonFromWKB(wkb[,srid])`
: Constrói um valor `POLYGON` usando sua representação WKB e SRID.

`MPointFromWKB(wkb[,srid])`
`MultiPointFromWKB(wkb[,srid])`
: Constrói um valor `MULTIPOINT` usando sua representação WKB e SRID.

`MLineFromWKB(wkb[,srid])`
`MultiLineStringFromWKB(wkb[,srid])`
: Constrói um valor `MULTILINESTRING` usando sua representação WKB e SRID.

`MPolyFromWKB(wkb[,srid])`
`MultiPolygonFromWKB(wkb[,srid])`
: Constrói um valor `MULTIPOLYGON` usando sua representação WKB e SRID.

`GeomCollFromWKB(wkb[,srid])`
`GeometryCollectionFromWKB(wkt[,srid])`
: Constrói um valor `GEOMETRYCOLLECTION` usando sua representação WKB e SRID.

A especificação do OpenGIS também descreve funções adicionais para construção de valores `Polygon` ou `MultiPolygon` baseados em uma representação WKB de uma coleção de anéis ou valores de `LineString` fechadas. Estes valores podem se interceptar. O MySQL ainda não implementou estas funções:

`BdPolyFromWKB(wkb,srid)`
: Constrói um valor `Polygon` a partir de um valor `MultiLineString` no formato WKB contendo uma coleção arbitrária de valores `LineString` fechados.

`BdMPolyFromWKB(wkb,srid)`
: Constrói um valor `MultiPolygon` a partir de um valor `MultiLineString` no formato WKB contendo uma coleção arbitrária de valores `LineString` fechados.

10.4.2.3 Criando uma Valor de Geometira Usando Funções Específicas do MySQL

Nota: o MySQL aindo não implementou as funções listadas nesta seção.

O MySQL fornece um conjunto de funções úteis para criar representações WKB de geometria. A função descrita nesta seção são extensões MySQL para a especificação OpenGIS.

O resultado destas funções são valores BLOBs contendo representações WKB de valores de geometria sem SRID. Os resultados destas funções podem ser substituidos como primeiro argumento para a família de funções GeomFromWKB().

Point(x,y)
: Constrói um Point WKB usando suas cooerdenadas.

MultiPoint(pt1,pt2,...)
: Constrói um MultiPoint WKB usando WKBPoints. Quando o argumento não é Point WKB, o valor de retorno é NULL.

LineString(pt1,pt2,...)
: Constrói um LineString WKB de um número de Points WKB. Quando o argumento não é Point WKB, o valor de retorno é NULL. Quando o número de Points é menor que dois o valor de retorno é NULL.

MultiLineString(WKBLineString,WKBLineString,...,WKBLineString)
: Constrói um MultiLineString WKB usando LineStrings WKB. Quando o argumento não é LineString WKB, o valor de retorno é NULL.

Polygon(ls1,ls2,...)
: Constrói um Polygon de um número de LineStrings WKB. Quando o arguemnto não representa o WKB de um LinearRing (ex. LineString não fechada e simples) o valor de retorno é NULL.

MultiPolygon(poly1,poly2,...)
: Constrói um MultiPolygon WKB de um conjunto de Polygons WKB. Quando o argumento não é um Polygon WKB, o valor de retorno é NULL.

GeometryCollection(WKBGeometry,WKBGeometry,..,WKBGeometry)
: Constucts a GeometryCollection WKB. Quando o argumento não é uma representação WKB bem formada de uma geometria, o valor de retorno é NULL.

10.4.3 Criando Colunas Espaciais

O MySQL fornece um modo padrão de criar colunas espaciais para tipos de geometria, por exemplo, com CREATE TABLE ou ALTER TABLE. Atualmente, colunas espaciais são suportadas apenas por tabelas MyISAM.

CREATE TABLE
: Use a instrução CREATE TABLE para criar uma tabela com uma coluna espacial:

```
mysql> CREATE TABLE geom (g GEOMETRY);
Query OK, 0 rows affected (0.02 sec)
mysql>
```

ALTER TABLE
: Use a instrução ALTER TABLE para adicionar ou deletar uma coluna espacial a ou de uma tabela existente:

```
mysql> ALTER TABLE geom ADD pt POINT;
Query OK, 0 rows affected (0.00 sec)
```

```
Records: 0  Duplicates: 0  Warnings: 0
mysql> ALTER TABLE geom DROP pt;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql>
```

10.4.4 Entrando com Dados em Colunas Espaciais

Depois de criar as colunas espaciais, você pode preenchê-las com os dados espaciais.

Os valores devem ser armazenados no formato de geometria interna, mas você pode convertê-las para este formato a partir dos formatos Well-Known Text (WKT) ou Well-Known Binary (WKB). Os exemplos a seguir demonstram como inserir valores de geometria em uma tabela convertendo valores WKT em formatos de geometria interna.

Você pode realizar a conversão diretamente na instrução `INSERT`:

```
INSERT INTO geom VALUES (GeomFromText('POINT(1 1)'));

SET @g = 'POINT(1 1)';
INSERT INTO geom VALUES (GeomFromText(@g));
```

Ou a coversão pode ser feita primeiro que o `INSERT`:

```
SET @g = GeomFromText('POINT(1 1)');
INSERT INTO geom VALUES (@g);
```

Os seguintes exemplos inserem geometrias mais comlexas nas tabelas:

```
SET @g = 'LINESTRING(0 0,1 1,2 2)';
INSERT INTO geom VALUES (GeomFromText(@g));

SET @g = 'POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))';
INSERT INTO geom VALUES (GeomFromText(@g));

SET @g = 'GEOMETRYCOLLECTION(POINT(1 1),LINESTRING(0 0,1 1,2 2,3 3,4 4))';
INSERT INTO geom VALUES (GeomFromText(@g));
```

Todos os exemplos anteiores usam `GeomFromText()` para criar os valores de geometria. Você também pode usar funções de tipo específicos:

```
SET @g = 'POINT(1 1)';
INSERT INTO geom VALUES (PointFromText(@g));

SET @g = 'LINESTRING(0 0,1 1,2 2)';
INSERT INTO geom VALUES (LineStringFromText(@g));

SET @g = 'POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))';
INSERT INTO geom VALUES (PolygonFromText(@g));

SET @g = 'GEOMETRYCOLLECTION(POINT(1 1),LINESTRING(0 0,1 1,2 2,3 3,4 4))';
INSERT INTO geom VALUES (GeomCollFromText(@g));
```

Note que se um programa aplicativo cliente que quiser utilizar representações WKB de valores de geometria, ele é responsável por enviar corretamente WKB formadas em consultas

para o servidor. No entanto, existem diversos modos de satisfazer esta exigência. Por exemplo:

- Inserindo um `Point(1,1)` com sintaxe literal hexa:

  ```
  INSERT INTO geom VALUES (GeomFromWKB(0x0101000000000000000000F03F000000000000F0
  ```

- Uma aplicação ODBC pode enviar uma representação WKB, ligando como um argumento do tipo BLOB:

  ```
  INSERT INTO geom VALUES (GeomFromWKB(?));
  ```

  Outra interfaces de programação podem suportar um mecanimo de placeholder similar.

- Em um programa C, você pode fazer um escape de um valor binário usando `mysql_real_escape_string()` e incluindo o resultado em string de consulta que é enviada ao servidor. Veja Seção 12.1.3.44 [`mysql_real_escape_string()`], Página 815.

10.4.5 Buscando Dados Espaciais

Valores de geometria, previamente armazenados na tabela, pode, ser buscados com a conversão em formatos internos. Você também pode convertê-los no formato WKT ou WKB.

10.4.5.1 Buscando Dados Espaciais em um Formato Interno

Buscar valores de geometria usando formatos internos pode ser útil em transferências de tabela para tabela:

```
CREATE TABLE geom2 (g GEOMETRY) SELECT g FROM geom;
```

10.4.5.2 Buscando Dados Espaciais no Formato WKT

A função `AsText()` fornece acesso textual a valores de geometria. Ele converte a geometria a partir de um formato interno em uma string WKT.

```
mysql> SELECT AsText(g) FROM geom;
+-------------------------+
| AsText(p1)              |
+-------------------------+
| POINT(1 1)              |
| LINESTRING(0 0,1 1,2 2) |
+-------------------------+
2 rows in set (0.00 sec)
```

10.4.5.3 Buscando Dados Espaciais no Formato WKB

A função `AsBinary` fornece acesso binário a valores de geometria. Ela converte uma geometria a partir de um formato interno em um `BLOB` contendo um valor WKB.

```
SELECT AsBinary(g) FROM geom;
```

10.5 Analisando Informação Espacial

Depois de preencher colunas espaciais com valores, você está pronto para consultá-los e analisá-los. O MySQL fornece um conjunto de funções para realizar diversas operações em dados espaciais. Estas funções podem ser agrupadas em quatro grandes categorias de acordo com o tipo de operação que eles realizam:

- Funções que convertem geometrias entre vários formatos.
- Funções que fornecem acesso a propriedades qualitativas ou quantitativas de um geometria
- Funções que descrevem realções entre duas geometrias.
- Funções que criam novas geometrias de outras existentes.

Funções de análise espacial podem ser usados em muitos contextos, tais como:

- Qualquer programa SQL interativo, como `mysql` ou `MySQLCC`.
- Aplicativos escritos em qualquer linguagem duportando uma API do cliente MySQL.

10.5.1 Funções Para Converter Geometrias Entre Formatos Diferentes

O MySQL suporta as seguintes funções para converter valores geométricos entre formatos internos e os formatos WKB e WKT:

`GeomFromText(wkt[,srid])`
: Converte um valor string de sua representação WKT em formato de geometria interna e retorna o resultado. Um número de funções específicas de tipo também são suportadas, como `PointFromText()` e `LineFromText()`; veja Seção 10.4.2.1 [GIS WKT Functions], Página 742.

`GeomFromWKB(wkb [,srid])`
: Converte um valor binário da sua representação WKB em formato de geometria interna e retorna o resultado. Um número de funções específicas de tipo também são suportadas, como `PointFromWKB()` e `LineFromWKB()`; veja Seção 10.4.2.2 [GIS WKB Functions], Página 743.

`AsText(g)`
: Converte um valor em formato de geomtria interna em sua representação WKT e retorna a string resultante.

```
mysql> SET @g = 'LineString(1 1,2 2,3 3)';
mysql> SELECT AsText(GeomFromText(@g));
+--------------------------+
| AsText(GeomFromText(@G)) |
+--------------------------+
| LINESTRING(1 1,2 2,3 3)  |
+--------------------------+
```

`AsBinary(g)`
: Converte um valor em formato de geomtria interna em sua representação WKB e retorna o valor binário resultante

10.5.2 Funções de Análise das Propriedades de `Geometry`

Cada função que pertencem a este grupo tomam um valor de geometria como seus argumentos e retornam alguma propriedade quantitativa e qualitativa desta geometria. Algumas funções restrigem os seus tipos de argumentos. tais funções retornam `NULL` se o argumento é de um tipo de geometria incorreta. Por exemplo, `Area()` retorna `NULL` se o tipo do objeto não for nem `Polygon` nem `MultiPolygon`.

10.5.2.1 Funções de Análise das Propriedades de `Geometry` em Geral

As funções listadas nesta seção não restrigem seus argumentos e acitam um valor geometria de qualquer tipo.

`GeometryType(g)`

Retorna como string o nome do tipo da geometria da qual esta instância `g` de geometry é um membro. O nome corresponderá a uma das subclasses instanciáveis de `Geometry`.

```
mysql> SELECT GeometryType(GeomFromText('POINT(1 1)'));
+---------------------------------------------+
| GeometryType(GeomFromText('POINT(1 1)')) |
+---------------------------------------------+
| POINT                                       |
+---------------------------------------------+
```

`Dimension(g)`

Retorna a dimensão herdada deste objeto `g` de geometria. O resultado pode ser −1, 0, 1 or 2. (o significado destes valores é dado em Seção 10.2.2 [GIS class geometry], Página 735.)

```
mysql> SELECT Dimension(GeomFromText('LineString(1 1,2 2)'));
+-----------------------------------------------+
| Dimension(GeomFromText('LineString(1 1,2 2)')) |
+-----------------------------------------------+
|                                             1 |
+-----------------------------------------------+
```

`SRID(g)` Retorna um inteiro indicando ID do Sistema de Referência Espacial do valor de geometria `g`.

```
mysql> SELECT SRID(GeomFromText('LineString(1 1,2 2)'));
+-------------------------------------------------+
| SRID(GeomFromText('LineString(1 1,2 2)',101)) |
+-------------------------------------------------+
|                                             101 |
+-------------------------------------------------+
```

`Envelope(geometry g):geometry`

Retorna o Retângulo de Limite Mínimo (Minimum Bounding Rectangle (MBR)) para o valor de geometria `g`. O resultado é retornado como um polygon (polígono).

```
mysql> SELECT AsText(Envelope(GeomFromText('LineString(1 1,2 2)',101)));
+-----------------------------------------------------+
| AsText(Envelope(GeomFromText('LineString(1 1,2 2)')) |
+-----------------------------------------------------+
| POLYGON((1 1,2 1,2 2,1 2,1 1))                      |
+-----------------------------------------------------+
```

O polygon é definido pelos pontos nos cantos da caixa que o limita:

```
POLYGON((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
```

A especificação OpenGIS também define as seguintes funções, que o MySQL ainda não implementou:

`Boundary(g)`
: Retorna uma geometria que é o fechamento do limite combinacional do valor da geometria `g`.

`IsEmpty(g)`
: Retorna 1 se o valor da geometria `g` e a geometria vazia, 0 se ela não está vazia e -1 se o argumento é `NULL`. Se a geometria está vazia, ela representa um conjunto de pontos vazios.

`IsSimple(g)`
: Atualmewnte esta função não deve ser usada. Quando implementada, seu comportamento será como descrito no próximo parágrafo.

  Retorna 1 se o valor da geometria `g` não tem nenhum ponto geométrico anormal, como a interseção própria ou tangente própria. `IsSimple` retorna 0 se o argumento não é simples, e -1 se é `NULL`.

  A descrição de cada geométrica instanciável dada anteriormente neste capítulo inclui a condição específica que faz com que uma instância desta classe seja classificada como não simples.

10.5.2.2 Funções de Análise das Propriedades de `Point`

Um `Point` consiste de suas coordenadas X e Y, que podem ser obtidas usando as seguintes funções:

`X(p)`
: Retorna o valor da coordenada X para o ponto `p` como um número de dupla precisão.

```
mysql> SELECT X(GeomFromText('Point(56.7 53.34)'));
+-------------------------------------+
| X(GeomFromText('Point(56.7 53.34)')) |
+-------------------------------------+
|                                56.7 |
+-------------------------------------+
```

`Y(p)`
: Retorna o valor da coordenada Y para o ponto `p` como um número de dupla precisão.

```
mysql> SELECT Y(GeomFromText('Point(56.7 53.34)'));
+-----------------------------------+
| Y(GeomFromText('Point(56.7 53.34)')) |
+-----------------------------------+
|                             53.34 |
+-----------------------------------+
```

10.5.2.3 Funções de Análise das Propriedades de `LineString`

Uma `LineString` consiste de valores `Point`. Você pode extrair pontos particulares de uma `LineString`, contar o número de pontos que ela contém ou obter o seu tamanho.

`EndPoint(ls)`

Retorna o `Point` que é o ponto final do valor `LineString` `ls`.

```
mysql> SELECT AsText(EndPoint(GeomFromText('LineString(1 1,2 2,3 3)')));
+-------------------------------------------------------------+
| AsText(EndPoint(GeomFromText('LineString(1 1,2 2,3 3)'))) |
+-------------------------------------------------------------+
| POINT(3 3)                                                  |
+-------------------------------------------------------------+
```

`GLength(ls)`

Returna como um número de precisão dupla o tamanho do valor `LineString` `ls` em sua referência espacial associada.

```
mysql> SELECT GLength(GeomFromText('LineString(1 1,2 2,3 3)'));
+-------------------------------------------------+
| GLength(GeomFromText('LineString(1 1,2 2,3 3)')) |
+-------------------------------------------------+
|                          2.8284271247462 |
+-------------------------------------------------+
```

`IsClosed(ls)`

Returna 1 se o valor `LineString` `ls` é fechado (isto é, seus valores `StartPoint()` e `EndPoint()` são os mesmos). Returna 0 se `ls` não é fechado, e -1 se ele é `NULL`.

```
mysql> SELECT IsClosed(GeomFromText('LineString(1 1,2 2,3 3)'));
+-------------------------------------------------+
| IsClosed(GeomFromText('LineString(1 1,2 2,3 3)')) |
+-------------------------------------------------+
|                                               0 |
+-------------------------------------------------+
```

`NumPoints(ls)`

retorna o número de pontos no valor `LineString` `ls`.

```
mysql> SELECT NumPoints(GeomFromText('LineString(1 1,2 2,3 3)'));
+-------------------------------------------------+
| NumPoints(GeomFromText('LineString(1 1,2 2,3 3)')) |
+-------------------------------------------------+
```

```
|                                                    3 |
+------------------------------------------------------+
```

`PointN(ls,n)`

Returna o `n`-ésimo ponto no valor `Linestring` `ls`.

```
mysql> SELECT AsText(PointN(GeomFromText('LineString(1 1,2 2,3 3)'),2));
+------------------------------------------------------------+
| AsText(PointN(GeomFromText('LineString(1 1,2 2,3 3)'),2)) |
+------------------------------------------------------------+
| POINT(2 2)                                                 |
+------------------------------------------------------------+
```

`StartPoint(ls)`

Returna o `Point` que é o ponto inicial do valor `LineString` `ls`.

```
mysql> SELECT AsText(StartPoint(GeomFromText('LineString(1 1,2 2,3 3)')))
+--------------------------------------------------------------+
| AsText(StartPoint(GeomFromText('LineString(1 1,2 2,3 3)'))) |
+--------------------------------------------------------------+
| POINT(1 1)                                                   |
+--------------------------------------------------------------+
```

A especificação OpenGIS também define as seguintes funções, que o MySQL ainda não implementou:

`IsRing(ls)`

Retorna 1 se o valor `LineString` `ls` é fechado (isto é, seus valores `StartPoinnt()` e `EndPoint()` são os mesmos) e é simples (não passa pelo mesmo ponto mais de uma vez). Retorna 0 se `ls` não é um anel, -1 se é `NULL`.

10.5.2.4 Funções de Análise das Propriedades de `MultiLineString`

`GLength(mls)`

Retorna o tamanho do valor de `MultiLineString` `mls` como um número e precisão dupla. O tamanha de `mls` é igual a soma dos tamanhos de seus elementos.

```
mysql> SELECT GLength(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5
+---------------------------------------------------------------------+
| GLength(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5 5))')) |
+---------------------------------------------------------------------+
|                                                 4.2426406871193 |
+---------------------------------------------------------------------+
```

`IsClosed(MultiLineString m):Integer`
`IsClosed(mls)`

Returna 1 se o valor `MultiLineString` `mls` é fechado (isto é, os valores `StartPoint()` e `EndPoint()` são os mesmos para cada `LineString` em `mls`). Returna 0 se `mls` não é fechada, e -1 se for `NULL`.

```
mysql> SELECT IsClosed(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5
+-----------------------------------------------------------------------+
```

```
| IsClosed(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5 5))')) |
+--------------------------------------------------------------------+
|                                                                  0 |
+--------------------------------------------------------------------+
```

10.5.2.5 Funções de Análise das Propriedades de `Polygon`

`Area(poly)`

Returna como um número de dupla precisão a área do valor `Polygon poly`, como medido em seu sistema de referência espacial.

```
mysql> SELECT Area(GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2
+---------------------------------------------------------------------
| Area(GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))
+---------------------------------------------------------------------
|
+---------------------------------------------------------------------
```

`NumInteriorRings(poly)`

Retorna o número de anéis interiores no valor `Polygon poly`.

```
mysql> SELECT NumInteriorRings(GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0
+---------------------------------------------------------------------
| NumInteriorRings(GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2
+---------------------------------------------------------------------
|
+---------------------------------------------------------------------
1 row in set (0.00 sec)
```

`ExteriorRing(poly)`

Retorna o anel exterior do valor `Polygon poly` como uma `LineString`.

```
mysql> SELECT AsText(ExteriorRing(GeomFromText('Polygon((0 0,0 3,3 3,3 0,
+---------------------------------------------------------------------
| AsText(ExteriorRing(GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1
+---------------------------------------------------------------------
| LINESTRING(0 0,0 3,3 3,3 0,0 0)
+---------------------------------------------------------------------
```

`InteriorRingN(poly,n)`

Retorna o `n`-ésimo anel exterior para o valor `Polygon poly` como uma `LineString`.

```
mysql> SELECT AsText(InteriorRingN(GeomFromText('Polygon((0 0,0 3,3 3,3 0
+---------------------------------------------------------------------
| AsText(InteriorRingN(GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1
+---------------------------------------------------------------------
| LINESTRING(1 1,1 2,2 2,2 1,1 1)
+---------------------------------------------------------------------
```

A especificação OpenGIS também define as seguintes funções, que o MySQL ainda não implementou:

`Centroid(poly)`
: O centóide matemático para o valor `Polygon` `poly` como um `Point`. O resultado não é garantido estar neste `Polygon`.

`PointOnSurface(poly)`
: Returna um valor `Point` que esta garantidamente no valor `Polygon` `poly`.

10.5.2.6 Funções de Análise das Propriedades de `MultiPolygon`

`Area(mpoly)`
: Retorna como um número de precisão dupla a área do valor `MultiPolygon` `mpoly`, como medido no sistema de referência espacial deste `MultiPolygon`.

```
mysql> SELECT Area(GeomFromText('MultiPolygon(((0 0,0 3,3 3,3 0,0 0),(1 1
+-----------------------------------------------------------------------
| Area(GeomFromText('MultiPolygon(((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1
+-----------------------------------------------------------------------
|
+-----------------------------------------------------------------------
```

A especificação OpenGIS também define as seguintes funções, que o MySQL ainda não implementou:

`Centroid(mpoly)`
: O centróide matemático para este `MultiPolygon` como um `Point`. O resultado não é garantido estar neste `MultiPolygon`.

`PointOnSurface(mpoly)`
: Retorna um valor `Point` que é garantido estar no valor `MultiPolygon` `mpoly`.

10.5.2.7 Funções de Análise das Propriedades de `GeometryCollection`

`NumGeometries(gc)`
: Retorna o número de geometrias no valor `GeometryCollection` `gc`.

```
mysql> SELECT NumGeometries(GeomFromText('GeometryCollection(Point(1 1),L
+-----------------------------------------------------------------------
| NumGeometries(GeomFromText('GeometryCollection(Point(1 1),LineString(2
+-----------------------------------------------------------------------
|
+-----------------------------------------------------------------------
```

`GeometryN(gc,n)`
: Retorna o `n`-ésima geometria no valor `GeometryCollection` `gc`. O número de geometrias começa em 1.

```
mysql> SELECT AsText(GeometryN(GeomFromText('GeometryCollection(Point(1 1
+-----------------------------------------------------------------------
| AsText(GeometryN(GeomFromText('GeometryCollection(Point(1 1),LineString
+-----------------------------------------------------------------------
```

```
| POINT(1 1)
+-----------------------------------------------------------------------
```

Nota: Funções para tipos de geometrias específicas retornam `NULL` se a geomtria passada é do tipo de geometria errado. Por exemplo `Area()` retorna `NULL` se o tipo do objeto não é nem Polygon nem `MultiPolygon`.

10.5.3 Funções Que Criam Novas Geometrias de Outras Existentes

10.5.3.1 Funções de Geometria Que Produzem Novas Geometrias

Na seção Seção 10.5.2 [Geometry property functions], Página 749 nós já discutimos algumas funções que podem construir novas geometrias se outras existentes:

- `Envelope(g)`
- `StartPoint(ls)`
- `EndPoint(ls)`
- `PointN(ls,n)`
- `ExteriorRing(poly)`
- `InteriorRingN(poly,n)`
- `GeometryN(gc,n)`

10.5.3.2 Operadores Espaciais

OpenGIS propõem algumas outras funções que podem produzir geometrias. Elas estão designadas a implementar Operadores Espaciais.

Estas funções ainda não estão implementadas no MySQL. Elas devem aparecer em distribuições futuras.

`Intersection(g1,g2)`
: Retorna uma geometria que representa a insterseção do conjunto de pontos dos valores das geometrias `g1` com `g2`.

`Union(g1,g2)`
: Retorna uma geometria que representa a união do conjunto de pontos dos valores das geometrias `g1` com `g2`.

`Difference(g1,g2)`
: Retorna uma geometria que representa a diferença do conjunto de pontos dos valores das geometrias `g1` com `g2`.

`SymDifference(g1,g2)`
: Retorna uma geometria que representa a diferença simétrica do conjunto de pontos dos valores das geometrias `g1` com `g2`.

`Buffer(g,d)`
: Retiorna uma geometria que representa todos os pontos cuja distância do valor da geometria `g` é menor que ou igual a distância de `d`.

`ConvexHull(g)`
: Retorna uma geometria que representa a casca convexa de do valor da geometria `g`.

10.5.4 Funções Para Testar Relações Espaciais Entre Objetos Geométricos

A função descrita nesta seção toma duas geometrias como parâmetros de entrada e retorna uma relação qualitativa ou quantitativa entre eles.

10.5.5 Relações de Retângulo de Limite Mínimo (Minimal Bounding Rectangles - MBR) em Geometrias

O MySQL fornece algumas funções que podem testar relações entre retângulos de limite mínimo de duas geometrias `g1` e `g2`. Elas incluem:

`MBRContains(g1,g2)`
: Retorna 1 ou 0 para indicar se o Retângulo de Limite Mínimo de `g1` contém o Retângulo de Limite Mínimo de `g2`.

```
mysql> SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
mysql> SET @g2 = GeomFromText('Point(1 1)');
mysql> SELECT MBRContains(@g1,@g2), MBRContains(@g2,@g1);
+----------------------+----------------------+
| MBRContains(@g1,@g2) | MBRContains(@g2,@g1) |
+----------------------+----------------------+
|                    1 |                    0 |
+----------------------+----------------------+
```

`MBRWithin(g1,g2)`
: Retorna 1 ou 0 para indicar se o Retângulo de Limite Mínimo de `g1` esta dentro do Retângulo de Limite Mínimo de `g2`.

```
mysql> SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
mysql> SET @g2 = GeomFromText('Polygon((0 0,0 5,5 5,5 0,0 0))');
mysql> SELECT MBRWithin(@g1,@g2), MBRWithin(@g2,@g1);
+--------------------+--------------------+
| MBRWithin(@g1,@g2) | MBRWithin(@g2,@g1) |
+--------------------+--------------------+
|                  1 |                  0 |
+--------------------+--------------------+
```

`MBRDisjoint(g1,g2)`
: Retorna 1 ou 0 para indicar se o Retângulo de Limite Mínimo de duas geometrias `g1` e `g2` não fazem interseção.

`MBREqual(g1,g2)`
: Retorna 1 ou 0 para indicar se o Retângulo de Limite Mínimo de duas geometrias `g1` e `g2` são o mesmo.

`MBRIntersects(g1,g2)`
: Retorna 1 ou 0 para indicar se o Retângulo de Limite Mínimo de duas geometrias `g1` e `g2` se interseptam.

`MBROverlaps(g1,g2)`
: Retorna 1 ou 0 para indicar se o Retângulo de Limite Mínimo de duas geometrias `g1` e `g2` se sobrepõe.

`MBRTouches(g1,g2)`
: Retorna 1 ou 0 para indicar se o Retângulo de Limite Mínimo de duas geometrias `g1` e `g2` se tocam.

10.5.6 Funções que Testam Relacionamentos Espaciais Entre Geometrias

A especificação OpenGIS define as seguintes funções, que o MySQL ainda não implementou. Elas devem aparecer em distribuições futuras. Quando implementadas, fornecerão suporte total para análise espacial, não apenas suporte baseado em MBR.

As funções operam em dois valores de geometria `g1` e `g2`.

`Contains(g1,g2)`
: Retorna 1 ou 0 para indicar se `g1` contem completamente `g2` ou não.

`Crosses(g1,g2)`
: Retorna 1 se `g1` cruza espacialmente `g2`. Retorna `NULL` se `g1` é um `Polygon` ou um `MultiPolygon`, ou se `g2` é um `Point` ou um `MultiPoint`. Senão 0 é retornado.

  O termo **spatially crosses** denota uma relação espacial entre duas geometrias que têm as seguintes propriedades:

  - As duas geometrias se interseptam
  - A interseção resulta em uma geometria que tem uma dimensão que é menor que a dimensão máxima das duas geometrias dadas.
  - A interseção não é igual a nenhuma das duas geometrias dadas.

`Disjoint(g1,g2)`
: Retorna 1 ou 0 para indicar se `g1` é espacialmente disjunta de `g2` ou não.

`Equals(g1,g2)`
: Retorna 1 ou 0 para indicar se `g1` é espacialmente igual a `g2` ou não.

`Intersects(g1,g2)`
: Retorna 1 ou 0 para indicar se `g1` intersepta espacialmente `g2` ou não.

`Overlaps(g1,g2)`
: Retorna 1 ou 0 para indicar se `g1` sobrepõe espacialmente a `g2` ou não. O termo `sobrepor espacialmente` é usado se duas geometrias fazem interseção e suas interseções resultam em uma geometria da mesma dimensão mas difernete de ambas as geometrias dadas.

`Touches(g1,g2)`
: Retorna 1 ou 0 para indicar se `g1` spatially touches `g2`, ou não. Duas geometrias se tocam espacialmente se o interiro de ambas geometrias não se interseptam, mas o limite de uma delas intersepta o limite ou o interior das geometrias.

`Within(g1,g2)`
: Retorna 1 ou 0 para indicar se `g1` está espacialmente dentro da `g2`, ou não.

`Distance(g1,g2)`
: Retorna como um número de precisão dupla, a menor distância entre quaiquer dois pontos nas duas geometrias.

`Related(g1,g2,pattern_matrix)`
: Retorna 1 ou 0 indicando se o relacionamento espacial especificado por `matriz_padrão` existe entre `g1` e `g2` ou não. Retorna -1 se os argumentos são `NULL`. A matriz padrão é uma string. Sua especificação será indicada aqui quando esta função estiver implementada.

10.6 Otimizando Análises Espaciais

É sabido que operações de busca em banco de dados não espaciais podem ser otimizadas utilizando índices. Isto ainda é verdade em banco de dados espaciais. Com a ajuda de grande variedades de métodos de indexacão multi-dimensionais, o quais já têm sido desenvolvidos, é possível otimizar buscas espaciais. As mais comuns delas são:

- Consulta de ponto que buscam por todos os objetos que contem um dado ponto.
- Consulta de região que buscam por todos os objetos que sobrepõe uma dada região.

O MySQL utiliza **Arvores R com separação quadrática** para indexar colunas espaciais. Um índice espacial é construído usando o MBR de uma geometria. Para a maioria da geometrias, o MBR é um retângulo mínimo que cerca a geometria. Para uma linha (linestring) horizontal ou uma vertical, o MBR é um retângulo degenerado, nas linhas e nos pontos respectivamente.

10.6.1 Criando Índices Espaciais

O MySQL pode criar índices espaciais usando uma sintaxe similar àquela usada para criar índices regulares, mas extendida com a palavra-chave `SPATIAL`. Colunas espaciais indexadas devem ser declaradas como `NOT NULL`. Os seguintes exemplos demonstram como criar índices de colunas espaciais.

- Com `CREATE TABLE`:

```
mysql> CREATE TABLE geom (g GEOMETRY NOT NULL, SPATIAL INDEX(g));
```

- Com `ALTER TABLE`:

```
mysql> ALTER TABLE geom ADD SPATIAL INDEX(g);
```

- Com `CREATE INDEX`:

```
mysql> CREATE SPATIAL INDEX sp_index ON geom (g);
```

Para remover índices espaciais, use `ALTER TABLE` ou `DROP INDEX`:

- Com `ALTER TABLE`:

```
mysql> ALTER TABLE geom (ADD SPATIAL KEY(g));
```

- Com `DROP INDEX`:

```
mysql> DROP INDEX sp_index ON geom;
```

Example: Suponha que uma tabela `geom` contém mais de 32000 geometrias, que estão armazenadas na coluna `g` do tipo `GEOMETRY`. A tabela também tem um campo `AUTO_INCREMENT` `fid`, armazenando valores dos IDs de objetos.

```
mysql> SHOW FIELDS FROM geom;
+-------+----------+------+-----+---------+----------------+
| Field | Type     | Null | Key | Default | Extra          |
+-------+----------+------+-----+---------+----------------+
| fid   | int(11)  |      | PRI | NULL    | auto_increment |
| g     | geometry |      |     |         |                |
+-------+----------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> SELECT COUNT(*) FROM geom;
+----------+
| count(*) |
+----------+
|    32376 |
+----------+
1 row in set (0.00 sec)
```

Para adicionar um índice espacial na coluna `g`, use esta instrução:

```
mysql> ALTER TABLE geom ADD SPATIAL INDEX(g);
Query OK, 32376 rows affected (4.05 sec)
Records: 32376  Duplicates: 0  Warnings: 0
```

10.6.2 Usando Índice Espacial

O otimizador investiga se os índices espaciais disponíveis podem ser envolvidos na busca se uma consulta com uma função como `MBRContains()` ou `MBRWithin()` na cláusula `WHERE` é executada. Por exemplo, suponhamos que queremos encontrar todos os objetos que estão no retângulo dado:

```
mysql> SELECT fid,AsText(g) FROM geom WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000
+-----+-------------------------------------------------------------------------
| fid | AsText(g)
+-----+-------------------------------------------------------------------------
|  21 | LINESTRING(30350.4 15828.8,30350.6 15845,30333.8 15845,30333.8 15828.8)
|  22 | LINESTRING(30350.6 15871.4,30350.6 15887.8,30334 15887.8,30334 15871.4)
|  23 | LINESTRING(30350.6 15914.2,30350.6 15930.4,30334 15930.4,30334 15914.2)
|  24 | LINESTRING(30290.2 15823,30290.2 15839.4,30273.4 15839.4,30273.4 15823)
|  25 | LINESTRING(30291.4 15866.2,30291.6 15882.4,30274.8 15882.4,30274.8 15866.2)
|  26 | LINESTRING(30291.6 15918.2,30291.6 15934.4,30275 15934.4,30275 15918.2)
```

```
| 249 | LINESTRING(30337.8 15938.6,30337.8 15946.8,30320.4 15946.8,30320.4 15938.4)
|   1 | LINESTRING(30250.4 15129.2,30248.8 15138.4,30238.2 15136.4,30240 15127.2)
|   2 | LINESTRING(30220.2 15122.8,30217.2 15137.8,30207.6 15136,30210.4 15121)
|   3 | LINESTRING(30179 15114.4,30176.6 15129.4,30167 15128,30169 15113)
|   4 | LINESTRING(30155.2 15121.4,30140.4 15118.6,30142 15109,30157 15111.6)
|   5 | LINESTRING(30192.4 15085,30177.6 15082.2,30179.2 15072.4,30194.2 15075.2)
|   6 | LINESTRING(30244 15087,30229 15086.2,30229.4 15076.4,30244.6 15077)
|   7 | LINESTRING(30200.6 15059.4,30185.6 15058.6,30186 15048.8,30201.2 15049.4)
|  10 | LINESTRING(30179.6 15017.8,30181 15002.8,30190.8 15003.6,30189.6 15019)
|  11 | LINESTRING(30154.2 15000.4,30168.6 15004.8,30166 15014.2,30151.2 15009.8)
|  13 | LINESTRING(30105 15065.8,30108.4 15050.8,30118 15053,30114.6 15067.8)
| 154 | LINESTRING(30276.2 15143.8,30261.4 15141,30263 15131.4,30278 15134)
| 155 | LINESTRING(30269.8 15084,30269.4 15093.4,30258.6 15093,30259 15083.4)
| 157 | LINESTRING(30128.2 15011,30113.2 15010.2,30113.6 15000.4,30128.8 15001)
+-----+--------------------------------------------------------------------------
20 rows in set (0.00 sec)
```

Agora verifiquemos o modo que esta consulta é executada, usando `EXPLAIN`:

```
mysql> EXPLAIN SELECT fid,AsText(g) FROM geom WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000
+----+-------------+-------+-------+---------------+------+---------+------+------+
| id | select_type | table | type  | possible_keys | key  | key_len | ref  | rows |
+----+-------------+-------+-------+---------------+------+---------+------+------+
|  1 | SIMPLE      | geom  | range | g             | g    |      32 | NULL |   50 |
+----+-------------+-------+-------+---------------+------+---------+------+------+
1 row in set (0.00 sec)
```

Agora verifiquemos o que aconteceria se nós não tivéssemos índices espaciais:

```
mysql> EXPLAIN SELECT fid,AsText(g) FROM geom IGNORE INDEX (g) WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000
+----+-------------+-------+------+---------------+------+---------+------+-------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows  |
+----+-------------+-------+------+---------------+------+---------+------+-------+
|  1 | SIMPLE      | geom  | ALL  | NULL          | NULL |    NULL | NULL | 32376 |
+----+-------------+-------+------+---------------+------+---------+------+-------+
1 row in set (0.00 sec)
```

Vamos executar a consulta acima, ignorando a chave espacial que temos:

```
mysql> SELECT fid,AsText(g) FROM geom IGNORE INDEX (g) WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000
+-----+--------------------------------------------------------------------------
| fid | AsText(g)
+-----+--------------------------------------------------------------------------
|   1 | LINESTRING(30250.4 15129.2,30248.8 15138.4,30238.2 15136.4,30240 15127.2)
|   2 | LINESTRING(30220.2 15122.8,30217.2 15137.8,30207.6 15136,30210.4 15121)
|   3 | LINESTRING(30179 15114.4,30176.6 15129.4,30167 15128,30169 15113)
|   4 | LINESTRING(30155.2 15121.4,30140.4 15118.6,30142 15109,30157 15111.6)
|   5 | LINESTRING(30192.4 15085,30177.6 15082.2,30179.2 15072.4,30194.2 15075.2)
```

```
|   6 | LINESTRING(30244 15087,30229 15086.2,30229.4 15076.4,30244.6 15077)
|   7 | LINESTRING(30200.6 15059.4,30185.6 15058.6,30186 15048.8,30201.2 15049.4)
|  10 | LINESTRING(30179.6 15017.8,30181 15002.8,30190.8 15003.6,30189.6 15019)
|  11 | LINESTRING(30154.2 15000.4,30168.6 15004.8,30166 15014.2,30151.2 15009.8)
|  13 | LINESTRING(30105 15065.8,30108.4 15050.8,30118 15053,30114.6 15067.8)
|  21 | LINESTRING(30350.4 15828.8,30350.6 15845,30333.8 15845,30333.8 15828.8)
|  22 | LINESTRING(30350.6 15871.4,30350.6 15887.8,30334 15887.8,30334 15871.4)
|  23 | LINESTRING(30350.6 15914.2,30350.6 15930.4,30334 15930.4,30334 15914.2)
|  24 | LINESTRING(30290.2 15823,30290.2 15839.4,30273.4 15839.4,30273.4 15823)
|  25 | LINESTRING(30291.4 15866.2,30291.6 15882.4,30274.8 15882.4,30274.8 15866.2)
|  26 | LINESTRING(30291.6 15918.2,30291.6 15934.4,30275 15934.4,30275 15918.2)
| 154 | LINESTRING(30276.2 15143.8,30261.4 15141,30263 15131.4,30278 15134)
| 155 | LINESTRING(30269.8 15084,30269.4 15093.4,30258.6 15093,30259 15083.4)
| 157 | LINESTRING(30128.2 15011,30113.2 15010.2,30113.6 15000.4,30128.8 15001)
| 249 | LINESTRING(30337.8 15938.6,30337.8 15946.8,30320.4 15946.8,30320.4 15938.4)
+-----+------------------------------------------------------------------------
20 rows in set (0.46 sec)
```

Quando o índice não é usado, o tempo de execução para esta consulta cresce de 0.00 segundos para 0.46 segundos.

Nas verões futuras, índices espaciais também serão usados para otimizar outras funções. Veja Seção 10.5.4 [Functions for testing spatial relations between geometric objects], Página 756.

10.7 Compatibilidade e Conformidade com o MySQL

10.7.1 Recursos GIS Que Ainda Não Estão Implementados

Views de Metadados Adicionais
: Especificações OpenGIS propõe várias views adicionais de metadados. Por exemplo, um sistema de view chamado `GEOMETRY_COLUMNS` contem uma descrição de colunas geometria, uma linha para coda coluna de geometria no banco de dados.

Funções para adicionar/apagar colunas espaciais
: OpenGIS assume que colunas podem ser adcionados ou apagados usando funções `AddGeometryColumn()` e `DropGeometryColumn()`. No MySQL isto deve ser feito utilizando as instruções `ALTER TABLE`, `CREATE INDEX` e `DROP INDEX`

Itens relacionados a Sistema de Referência Espaciasi e suas IDs (SRIDs):

- Funções como `Length()` e `Area()` assumem um sistemas de coordenadas planas.
- Todos os objetos são ataulmente considerados como estando no mesmo sistema de coordenadas planas.

A função OpenGIS `Length()` em `LineString` e `MultiLineString`
atualmente devem ser chamadas como `GLength()` no MySQL. O problema é que ela conflita com a função SQL existente `Length()` que calcula o tamanho de um valor string e algumas vezes não é possível distinguir se a função foi chamada co contexto textual ou espacial. Nós precisamos resolver isto de algum modo, ou escolher um outro nome de função.

11 Stored Procedures e Funções

Stored procedures e funções são recursoso novos no MySQL versão 5.0. Uma stored procedure é um conjunto de comandos SQL que podem ser armazenados no servidor. Uma vez que isto tenha sido feito, os clientes não precisam de reenviar os comnados individuais mas pode fazer referência às stored procedures.

Stored procedures podem fornecer um aumento no desempenho já que menos informação precisa ser enviada entre o servidor e o cliente. O lado negativo é que isto aumenta a carga no sistema do servidor de banco de dados, já que a maior parte do trabalho é feita no servidor e menor parte é feita do lado do cliente (aplicação). E geralmente existem muitas máquinas clientes (como servidoes web) mas apenas um ou poucos servidores e banco de dados.

Stored procedures também permitem que você tenha bibliotecas de funções no servidor de banco de dados. No entanto, linguagens de aplicações modernas já permitem que isto seja feito internamente com classes, por exemplo, e usar estes recursos das linguagens de aplicações clientes é benéfico para o programador mesmo fora do escopo do banco de dados usado.

Situações onde stored procedures fazem sentido:

- Quando várias aplicações clientes são escritas em diferentes linguagens ou funcionam em diferentes plataformas, mas precisam realizar as mesmas operações de banco de dados.
- Quando a segurança é prioritária. Bancos, por exemplo, usam stored procedures para todas as operações comuns. Isto fornece um ambiente consistente e seguro, e procedures podem assegurar que cada operação seja registrada de forma apropriada. Neste tipo de condiguração, aplicações e usuários não conseguiriam nenhuma acesso as tabelas do banco de dados diretamente, mas apenas podem executar stored procedures específicas.

O MySQL segue a sintaxe SQL:2003 para stored procedures, que também é usada pelo DB2 da IBM. Suporte para compatibilidade de outras linguagens de stored procedures (PL/SQL, T-SQL) podem ser adicionadas posteriormente.

A implementação do MySQL de stored procedures ainda está em progresso. Todas as sintaxes descritas neste capítulo são suportadas e qualquer limitação e extensão está documentada de forma aprorpiada.

Stored procedures exigem a tabela `proc` na banco de dados `mysql`. Esta tabela é criada durante a instalação do MySQL 5.0. Se você ataulizar para o MySQL 5.0 a partir de uma versão anterior, certifique de atualizar a sua tabela de permissão para ter certeza que a tabela `proc` existe. Veja Seção 2.5.6 [Upgrading-grant-tables], Página 130.

11.1 Sintaxe de Stored Procedure

Stored procedures e funções são rotinas criadas com as instruções `CREATE PROCEDURE` e `CREATE FUNCTION`. Um procedimento é chamado usando uma instrução `CALL` e só pode passar valores de retorno usadno variáveis de saída. Funções podem retornar um valor escalar e pode ser chamadas de dentro de uma instrução como quqlquer outra função (isto é, chamando o nome da funcao). Rotinas armazenadas podem chamar outras rotinas armazenadas. Uma rotina pode ser tanto um procedimento como uma função.

Atualmentem o MySQL só preserva o contexto para o banco de dados padrão. Isto é, se você usar `USE dbname` dentro de um procedimento, o banco de dados original é restaurado depois da saída da rotina. Uma rotina herda o banco de dados padrão de quem a chama, assim geralmente as rotinas devem utilizar uma instrução `USE dbname`, ou especifique todas as tabelas com uma referência de banco de dados explicita, ex. `dbname.tablename`.

O MySQL suporta uma extensão muito útil que permite o uso da instrução regular `SELECT` (isto é, sem usar cursores ou variáveis locais) dentro de uma stored procedure. O resultado de tal consulta é simplesmente enviado diretamente para o cliente. Várias instruções `SELECT` geram vária resultados, assim o cliente deve usar um biblioteca cliente do MySQL que suporta vários resultados. Isto significa que o cliente deve usar uma biblioteca cliente a partir de uma versão do MySQL mais recente que 4.1, pelo menos.

A seção seguinte descreve a sintaxe usada para criar, alterar, remover e condultar stored procedures e funções.

11.1.1 Manutenção de Stored Procedures

11.1.1.1 CREATE PROCEDURE e CREATE FUNCTION

```
CREATE PROCEDURE sp_name ([parameter[,...]])
[characteristic ...] routine_body

CREATE FUNCTION sp_name ([parameter[,...]])
[RETURNS type]
[characteristic ...] routine_body

parameter:
  [ IN | OUT | INOUT ] param_name type

type:
  Any valid MySQL data type

characteristic:
    LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | SQL SECURITY {DEFINER | INVOKER}
  | COMMENT string

routine_body:
  Valid SQL procedure statement(s)
```

A cláusula `RETURNS` pode ser especificada apenas por uma `FUNCTION`. É usada para indicar o tipo de retorno da função, e o corpo da função deve conter uma instrução `RETURN value`.

A lista de parâmetros entre parenteses deve estar sempre presente. Se não houver parâmetros, uma lista de parâmetros vazia de `()` deve ser usada. Cada parâmetro é um parâmetro `IN` por padrão. Para especificar outro tipo de parâmetro, use a palavra chave

`OUT` ou `INOUT` antes do nome do parâmetro. Especificar `IN`, `OUT` ou `INOUT` só é valido para uma `PROCEDURE`.

A instrução `CREATE FUNCTION` é usada em versão novas do MySQL para suporte a UDFs (User Defined Functions - Funções Definidas pelo Usuário). Veja Seção 14.2 [Adding functions], Página 899. As UDFs continuam a ser suportadas, mesmo com a existencia de stored functions. Uma UDF pode ser considerada como uma stored function externa. No entanto, note que stored functions compartilham os seus namespace com as `UDF`s.

Um framework para stored procedures externas serão introduzidas em um futuro próxima. Isto permitirá que você escreva stored procedures em outras linguagens além de SQL. Provavelmente, uma das primeiras linguagens a ser suportada sea PHP, já que o mecanismo do PHP é pequeno, seguro com threads e pode facilmente ser embutido. Como o framework será publico, é esperado que muitas outras linguagens também sejam suportadas.

Uma função é considerada "deterministica" se ela sempre retorna o mesmo resultado para os mesmos parâmetros de entrada, e "não deterministica" caso contrário. O otimizado pode usar este fato. Atualmente, a característica `DETERMINISTIC` é aceita, mas ainda não é usada.

A característica `SQL SECURITY` pode ser usada para especificar se a rotina deve ser executada usando as permissões do usuário que criou a rotina, ou o usuário que a chamou. O valor padrão é `DEFINER`. Este recurso é novo no SQL:2003.

O MySQL ainda não usa o privilégio `GRANT EXECUTE`. Assim ,por enquanto, se um procedimento `p1()` chama a tabela `t1`,o usuário deve ter privilégios na tabela `t1` para chamar o procedimento `p1()` com sucesso.

MySQL stores the `SQL_MODE` settings in effect at the time a routine is created, and will always execute routines with these settings in force.

A cláusula `COMMENT` é uma extensão do MySQL, e pode ser usada para descrever o stored procedure. Esta informação é exibida pelas instruções `SHOW CREATE PROCEDURE` e `SHOW CREATE FUNCTION`.

O MySQL permite rotinas contendo instruções DDL (como `CREATE` e `DROP`) e instruções de transação SQL (como `COMMIT`). Isto não é exigido por padrão e depende de especificações de implementação.

NOTA: Atualmente, stored `FUNCTION`s não podem conter referências às tabelas. Note que isto inclui algumas instruções `SET`, mas exclui algumas instruções `SELECT`. Esta limitação será retirada assim que possível.

A seguir temos um exemplo de uma stored procedure simples que usa um parâmetro `OUT`. O exemplo usa o comando `delimiter` do cliente `mysql` para alterar o delimitador de instrução para antes da definição do procedure. Isto permite que o delimitador '`;`' usado no corpo de procedure seja passado para o servidor em vez de ser interpretado pelo `mysql`.

```
mysql> delimiter |

mysql> CREATE PROCEDURE simpleproc (OUT param1 INT)
    -> BEGIN
    ->   SELECT COUNT(*) INTO param1 FROM t;
    -> END
    -> |
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> CALL simpleproc(@a)|
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @a|
+------+
| @a   |
+------+
| 3    |
+------+
1 row in set (0.00 sec)
```

A seguir esta um exemplo de uma função que utiliza um parametro, realiza uma operação usando uma função SQL e retorna o resultado:

```
mysql> delimiter |

mysql> CREATE FUNCTION hello (s CHAR(20)) RETURNS CHAR(50)
    -> RETURN CONCAT('Hello, ',s,'!');
    -> |
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT hello('world')|
+----------------+
| hello('world') |
+----------------+
| Hello, world!  |
+----------------+
1 row in set (0.00 sec)
```

11.1.1.2 ALTER PROCEDURE e ALTER FUNCTION

```
ALTER PROCEDURE | FUNCTION sp_name [characteristic ...]

characteristic:
    NAME newname
  | SQL SECURITY {DEFINER | INVOKER}
  | COMMENT string
```

Este comando pode ser usado para renomear uma stored procedure ou function, e para alterar suas características. Mais de uma mudança pode ser especificada em uma instrução `ALTER PROCEDURE` ou `ALTER FUNCTION`.

11.1.1.3 DROP PROCEDURE e DROP FUNCTION

```
DROP PROCEDURE | FUNCTION [IF EXISTS] sp_name
```

Este comando é usado para deletar uma stored procedure ou function. Isto é, a rotina especificada é removida do servidor.

A cláusula `IF EXISTS` é uma extensão do MySQL. Ela previne que um erro ocorra se o procedimento ou função não existe. Um aviso é produzido e pode ser vizualizado com `SHOW WARNINGS`.

11.1.1.4 `SHOW CREATE PROCEDURE` e `SHOW CREATE FUNCTION`

```
SHOW CREATE PROCEDURE | FUNCTION sp_name
```

Este comando é uma extensão do MySQL. De forma similar a `SHOW CREATE TABLE`, ele retorna a string exata que pode ser usada para recriar a rotina chamada.

11.1.2 `SHOW PROCEDURE STATUS` e `SHOW FUNCTION STATUS`

```
SHOW PROCEDURE | FUNCTION STATUS [LIKE pattern]
```

Este comando é uma extensão do MySQL. Ele retorna características da rotina, tais como nome, tipo, quem criou, datas de modificação e criação. Se nenhum padrão é especificado, a informação de todas as stored procedures ou todas as stored functions é listado, dependendo de qual instrução você utiliza.

11.1.3 `CALL`

```
CALL sp_name([parameter[,...]])
```

O comando `CALL` é usado para chamar uma rotina que foi definida anteriormente com `CREATE PROCEDURE`.

11.1.4 `BEGIN ... END` Compound Statement

```
[begin_label:] BEGIN
 statement(s)
END [end_label]
```

As rotinas armazenadas podem conter várias instruções, usando um instrução `BEGIN ... END`.

`begin_label` e `end_label` devem ser os mesmos, se ambos forem especificados.

Notem que a cláusula opcional `[NOT] ATOMIC` ainda não é suportada. Isto significa que nenhum savepoint de transação é definido no início do bloco da instrução e a cláusula `BEGIN` usada neste contexto não tem nenhum efeito no transação atual.

Várias instruções exigem que um cliente tenha permissão para enviar strings de querys contendo '`;`'. Isto é tratado no cliente mysql e linha de comando com o comando `delimiter`. Alterando o delimitador '`;`' do final da consulta (por exemplo, para '`|`') permite que '`;`' seja usado no corpo de uma rotina.

11.1.5 Instrução `DECLARE`

A instrução `DECLARE` é usada para definir vários itens locais para uma rotina: variaveis locais (veja Seção 11.1.6 [Variables in Stored Procedures], Página 768), condições e handlers (veja

Seção 11.1.7 [Conditions and Handlers], Página 768) e cursors (veja Seção 11.1.8 [Cursors], Página 770). As instruções `SIGNAL` e `RESIGNAL` ainda não são suportadas.

`DECLARE` só pode ser usada dentro de uma instrução composta `BEGIN ... END` e deve estar no início, antes de qualquer outra instrução.

11.1.6 Variables in Stored Procedures

Você pode declarar e usar variáveis dentro de uma rotina.

11.1.6.1 Variável Local `DECLARE`

```
DECLARE var_name[,...] type [DEFAULT value]
```

Este comando é usado para declarar variáveis locais. O escopo de uma variável está dentro do bloco `BEGIN ... END`.

11.1.6.2 Instrução Variável `SET`

```
SET variable = expression [,...]
```

A instrução `SET` em stored procedures é uma versão estendida do comando `SET` geral. As variáveis indicadas podem ser aquelas declaradas dentro de uma rotina, ou variáveis globais do servidor.

A instrução `SET` em stored procedures é implementada como parte da sintaxe pré-existente de `SET`. Isto permite uma sintaxe estendida de `SET a=x, b=y, ...` onde variáveis de tipos diferentes (variaveis declaradas localamente, variáveis do servidor e variáveis globais e de sessão do servidor) podem estar misturadas. Ela também permite combinações de variáveis locais e algumas opções que só fazem sentido para variáveis globais e de sistema; neste caso as opções são aceitas mas ignoradas.

11.1.6.3 Instrução `SELECT ... INTO`

```
SELECT column[,...] INTO variable[,...] table_expression
```

Esta sintaxe de `SELECT` armazena colunas selecionadas diretamente nas variáveis. Por esta razão, apenas uma linha pode ser recuperada. Esta instrução também é extremamente útil quando usada em conjunto com cursores.

```
SELECT id,data INTO x,y FROM test.t1 LIMIT 1;
```

11.1.7 Condições e Handlers

Certas condições podem exigir tratamento específico. Estas condições podem ser relacionadas a erros, bem como controle de fluxo geral dentro da rotina.

11.1.7.1 `DECLARE` Conditions

```
DECLARE condition_name CONDITION FOR condition_value
```

```
condition_value:
    SQLSTATE [VALUE] sqlstate_value
  | mysql_error_code
```

Esta instrução especifica condições que necessitarão de tratamento especial. Ela associa um nome com uma condição de erro específica. O nome pode ser subsequentemente usado em uma instrução `DECLARE HANDLER`. Veja Seção 11.1.7.2 [DECLARE Handlers], Página 769.

Além dos valores SQLSTATE, códigos de erro do MySQL também são suportados.

11.1.7.2 DECLARE Handlers

```
DECLARE handler_type HANDLER FOR condition_value[,...] sp_statement

handler_type:
    CONTINUE
  | EXIT
  | UNDO

condition_value:
    SQLSTATE [VALUE] sqlstate_value
  | condition_name
  | SQLWARNING
  | NOT FOUND
  | SQLEXCEPTION
  | mysql_error_code
```

Esta instrução especifica handlers para lidar com uma ou mais condições. Se uma dessas condições ocorrer, a instrução especificada é executada.

Para um handler `CONTINUE`, a execução das rotinas atuais continuam depois da instrução handler. Para um handler `EXIT`, a execução da rotina atual é terminada. O `handler_type UNDO` ainda não é suportado. Atualmente o `UNDO` se comporta como `CONTINUE`.

`SQLWARNING` is shorthand for all SQLSTATE codes that begin with 01.

`NOT FOUND` is shorthand for all SQLSTATE codes that begin with 02.

`EXCEPTION` is shorthand for all SQLSTATE codes not caught by `SQLWARNING` or `NOT FOUND`.

Além dos valores SQLSTATE, códigos de erro do MySQL também são suportados.

Por exemplo:

```
mysql> CREATE TABLE test.t (s1 int,primary key (s1));
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter |

ysql> CREATE PROCEDURE handlerdemo ()
    -> BEGIN
    ->   DECLARE CONTINUE HANDLER FOR '23000' SET @x2 = 1;
    ->   set @x = 1;
```

```
    ->   INSERT INTO test.t VALUES (1);
    ->   set @x = 2;
    ->   INSERT INTO test.t VALUES (1);
    ->   SET @x = 3;
    -> END;
    -> |
Query OK, 0 rows affected (0.00 sec)

mysql> CALL handlerdemo()|
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @x|
    +------+
    | @x   |
    +------+
    | 3    |
    +------+
    1 row in set (0.00 sec)
```

Notice that `@x` is 3, which shows that MySQL executed to the end of the procedure. If the line `DECLARE CONTINUE HANDLER FOR '23000' SET @x2 = 1;` had not been present, MySQL would have taken the default (`EXIT`) path after the second `INSERT` failed due to the `PRIMARY KEY` constraint, and `SELECT @x` would have returned 2.

11.1.8 Cursors

Simple cursors are supported inside stored procedures and functions. The syntax is as in embedded SQL. Cursors are currently asensitive, read-only, and non-scrolling. Asensitive means that the server may or may not make a copy of its result table.

For example:

```
CREATE PROCEDURE curdemo()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
  DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
  DECLARE a CHAR(16);
  DECLARE b,c INT;

  OPEN cur1;
  OPEN cur2;

  REPEAT
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF NOT done THEN
       IF b < c THEN
```

```
          INSERT INTO test.t3 VALUES (a,b);
        ELSE
          INSERT INTO test.t3 VALUES (a,c);
        END IF;
      END IF;
    UNTIL done END REPEAT;

    CLOSE cur1;
    CLOSE cur2;
  END
```

11.1.8.1 Declaring Cursors

```
DECLARE cursor_name CURSOR FOR sql_statement
```

Multiple cursors may be defined in a routine, but each must have a unique name.

11.1.8.2 Cursor `OPEN` Statement

```
OPEN cursor_name
```

This statement opens a previously declared cursor.

11.1.8.3 Cursor `FETCH` Statement

```
FETCH cursor_name
```

This statement fetches the next row (if a row exists) using the specified open cursor, and advances the cursor pointer.

11.1.8.4 Cursor `CLOSE` Statement

```
CLOSE cursor_name
```

This statement closes a previously opened cursor.

11.1.9 Flow Control Constructs

The `IF`, `CASE`, `LOOP`, `WHILE`, `ITERATE`, and `LEAVE` constructs are fully implemented.

These constructs may each contain either a single statement, or a block of statements using the `BEGIN ... END` compound statement. Constructs may be nested.

`FOR` loops are not currently supported.

11.1.9.1 `IF` Statement

```
IF search_condition THEN statement(s)
[ELSEIF search_condition THEN statement(s)]
...
[ELSE statement(s)]
```

```
END IF
```

`IF` implements a basic conditional construct. If the `search_condition` evaluates to true, the corresponding SQL statement is executed. If no `search_condition` matches, the statement in the `ELSE` clause is executed.

Please note that there is also an `IF()` function. Veja Seção 6.3.1.4 [Control flow functions], Página 510.

11.1.9.2 `CASE` Statement

```
CASE case_value
  WHEN when_value THEN statement
  [WHEN when_value THEN statement ...]
  [ELSE statement]
END CASE
```

or

```
CASE
  WHEN search_condition THEN statement
  [WHEN search_condition THEN statement ...]
  [ELSE statement]
END CASE
```

`CASE` implements a complex conditional construct. If a `search_condition` evaluates to true, the corresponding SQL statement is executed. If no search condition matches, the statement in the `ELSE` clause is executed.

Please note that the syntax of a `CASE` statement inside a stored procedure differs slightly from that of the SQL `CASE` expression. The `CASE` statement can not have an `ELSE NULL` clause, and the construct is terminated with `END CASE` instead of `END`. Veja Seção 6.3.1.4 [Control flow functions], Página 510.

11.1.9.3 `LOOP` Statement

```
[begin_label:] LOOP
  statement(s)
END LOOP [end_label]
```

`LOOP` implements a simple loop construct, enabling repeated execution of a particular statement or group of statements. The statements within the loop are repeated until the loop is exited, usually this is accomplished with a `LEAVE` statement.

`begin_label` and `end_label` must be the same, if both are specified.

11.1.9.4 `LEAVE` Statement

```
LEAVE label
```

This statement is used to exit any flow control construct.

11.1.9.5 ITERATE Statement

```
ITERATE label
```

ITERATE can only appear within LOOP, REPEAT, and WHILE statements. ITERATE means "do the loop iteration again."

For example:

```
CREATE PROCEDURE doiterate(p1 INT)
BEGIN
  label1: LOOP
    SET p1 = p1 + 1;
    IF p1 < 10 THEN ITERATE label1; END IF;
    LEAVE label1;
  END LOOP label1;
  SET @x = p1;
END
```

11.1.9.6 REPEAT Statement

```
[begin_label:] REPEAT
  statement(s)
UNTIL search_condition
END REPEAT [end_label]
```

The statements within a REPEAT statement are repeated until the search_condition is true.

begin_label and end_label must be the same, if both are specified.

For example:

```
mysql> delimiter |

mysql> CREATE PROCEDURE dorepeat(p1 INT)
    -> BEGIN
    ->   SET @x = 0;
    ->   REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
    -> END
    -> |
Query OK, 0 rows affected (0.00 sec)

mysql> CALL dorepeat(1000)|
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @x|
+------+
| @x   |
+------+
| 1001 |
+------+
1 row in set (0.00 sec)
```

11.1.9.7 WHILE Statement

```
[begin_label:] WHILE search_condition DO
  statement(s)
END WHILE [end_label]
```

The statements within a `WHILE` statement are repeated as long as the `search_condition` is true.

`begin_label` and `end_label` must be the same, if both are specified.

For example:

```
CREATE PROCEDURE dowhile()
BEGIN
  DECLARE v1 INT DEFAULT 5;

  WHILE v1 > 0 DO
    ...
    SET v1 = v1 - 1;
  END WHILE;
END
```

12 Ferramentas de Clientes e APIs do MySQL

Este capítulo descreve as APIs disponíveis para o MySQL, onde consegui-las e como utilizá-las. A API C é a coberta mais estensamente, já que ela foi desenvolvida pela equipe do MySQL e é a base para a maioria das outras APIs.

12.1 API C do MySQL

O código da API C é distribuído com o MySQL. Ele está incluído na biblioteca `mysqlclient` e permite programas em C a fazer acesso em banco de dados.

Muitos dos clientes na distribuição fonte do MySQL está escrito em C. Se você estiver procurando por exemplos que demonstrem como utilizar a API C, dê uma olhada neste clientes. Você pode encontrá-los no diretório `clients` na distribuição fonte do MySQL.

A maioria das outras clientes API (todos exceto Connector/J) usam a biblioteca `mysqlclient` para se comunicar com o servidor MySQL. Isto significa que, por exemplo, você pode tirar vantagem das mesmas variáveis de ambientes que são utilizados por outros programas clientes, pois eles referênciados pela biblioteca. Veja Seção 4.9 [Client-Side Scripts], Página 345, para uma lista destas variáveis.

O cliente tem um tamanho máximo de buffer de comunicação. O tamanho do buffer que é alocado inicialmente (16K bytes) é automaticamente aumentado para o tamanho máximo (o máximo é 16M). Como o tamanho do buffer é aumentado somente como autorização de demanda, o simples aumento do limite máximo padrão não faz, por si só, que mais recursos sejam usado. Esta verificação de tamanho é na maioria verificações por consultas erradas e pacotes de comunicações.

O buffer de comunicação deve ser grande o suficiente para conter uma única instrução SQL (para tráfego cliente-servidor) e uma linha de dado retornado (para trafico servidor-cliente). Cada buffer de comunicação de thread é dinamicamente aumentado para manipular qualquer consulta ou linha até o limite máximo. Por exemplo, se você tiver valores `BLOB` que contenham até 16M de dados, você deve ter um limite de buffer de comunicação de pelo menos 16M (no servidor e no cliente). A máximo padrão do cliente '16M. mas o máximo padrão no servidor é 1M. Você pode aumentar iso alterando o valor do parâmetro `max_allowed_packet` quando o servidor é iniciado. Veja Seção 5.5.2 [Par6ametros de servidor], Página 454.

O servidor MySQL encolhe cada buffer de comunicação para `net_buffer_length` bytes depois de cada consulta. Para clientes, o tamanho do buffer associado com um conexão não é reduzido até que a conexão seja fechada, quando a memória de tempo do cliente é recuperada.

Para programação com threads, veja Seção 12.1.14 [Threaded clients], Página 863. Para criar uma aplicação stand-alone que inclua o "servidor" e o "cliente" no mesmo programa (e que não comunica com um servidor MySQL externo), veja Seção 12.1.15 [libmysqld], Página 864.

12.1.1 Tipos de Dados da API C

`MYSQL` Esta estrutura representa um manpulador para uma conexão ao banco de dados. É usada para quase todas as funções MySQL.

`MYSQL_RES`
Esta estrutura representa o resultado de uma consulta que retorna linhas (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`). A informação retornada de uma consulta é chamada *conjunto de resultado* no resto desta seção.

`MYSQL_ROW`
Esta é uma representação segura de tipo de uma linha de dados. Ela é implementada atualmente como um vetor de strings de tamanho fixo (Você não pode tratá-los como strings terminadas com null se os valores do campo podem conter dados binários, porque tais valores podem conter um byte null internamente.). Linhas são obtidas pela chamada de `mysql_fetch_row()`.

`MYSQL_FIELD`
Esta estrutura contém informação sobre um campo, tais como nome, tipo e tamanho do campo. Seus membros são descritos em mais detalhes aqui. Você pode obter a estrutura `MYSQL_FIELD` para cada campo chamando `mysql_fetch_field()` repetidamente. Valores de campos não são parte desta estrutura; eles estão contidos na estrutura `MYSQL_ROW`.

`MYSQL_FIELD_OFFSET`
Esta é uma representação segura de um offset em uma lista de campos MySQL. (Usado por `mysql_field_seek()`.) Offsets são números de campos em um registro, começando com zero.

`my_ulonglong`
O tipo usado pelo número de linhas e para `mysql_affected_rows()`, `mysql_num_rows()`, e `mysql_insert_id()`. Este tipo fornece uma faixa de `0` a `1.84e19`.

Em alguns sistemas, tentar imprimir um valor do tipo `my_ulonglong` não funcionará. Para imprimir tais valores, converta-os para `unsigned long` e use o formato de impressão `%lu`. Exemplo:

```
printf ("Número de linhas: %lu\n", (unsigned long) mysql_num_rows(resulta
```

A estrutura `MYSQL_FIELD` contem os membros listados aqui:

`char * name`
O nome do campo, como um string terminada com null.

`char * table`
O nome da tabela contendo este campo, se não for um campo calculado. Para campos calculador, o valor `table` é uma string vazia.

`char * def`
O valor padrão para este campo, como um string terminada em null. Ele é atribuido apenas se você utilizar `mysql_list_fields()`.

`enum enum_field_types tipo`

O tipo do campo. O valor `tipo` pode ser um dos seguintes:

Valou tipo	Descrição do tipo
`FIELD_TYPE_TINY`	campo `TINYINT`
`FIELD_TYPE_SHORT`	campo `SMALLINT`
`FIELD_TYPE_LONG`	campo `INTEGER`
`FIELD_TYPE_INT24`	campo `MEDIUMINT`
`FIELD_TYPE_LONGLONG`	campo `BIGINT`
`FIELD_TYPE_DECIMAL`	campo `DECIMAL` ou `NUMERIC`
`FIELD_TYPE_FLOAT`	campo `FLOAT`
`FIELD_TYPE_DOUBLE`	campo `DOUBLE` ou `REAL`
`FIELD_TYPE_TIMESTAMP`	campo `TIMESTAMP`
`FIELD_TYPE_DATE`	campo `DATE`
`FIELD_TYPE_TIME`	campo `TIME`
`FIELD_TYPE_DATETIME`	campo `DATETIME`
`FIELD_TYPE_YEAR`	campo `YEAR`
`FIELD_TYPE_STRING`	campo `CHAR`
`FIELD_TYPE_VAR_STRING`	campo `VARCHAR`
`FIELD_TYPE_BLOB`	campo `BLOB` ou `TEXT` (usa `max_length` para determinar o tamanho máximo)
`FIELD_TYPE_SET`	campo `SET`
`FIELD_TYPE_ENUM`	campo `ENUM`
`FIELD_TYPE_NULL`	campo tipo-`NULL`
`FIELD_TYPE_CHAR`	Deprecado; use `FIELD_TYPE_TINY`

Você pode utilizar a macro `IS_NUM()` para testar se uma campo tem um tipo numérico. Passe o valor `tipo` para `IS_NUM()` e ele irá avaliar como VERDADEIRO (TRUE) se o campo for numérico:

```
if (IS_NUM(campo->tipo))
    printf("Campo é numérico\n");
```

`unsigned int length`

A largura de um campo, como especificado nas definições da tabela.

`unsigned int max_length`

A largura máxima do campo no conjunto de resultados (O tamanho do maior valor do campo para os registro no resultado atual). Se você utilizar `mysql_store_result()` ou `mysql_list_fields()`, ele contem o tamanho máximo para o campo. Se você utiliza `mysql_use_result()`, o valor desta variável é zero.

`unsigned int param`

Diferentes parâmetros binários para o campo. O valor de `param` pode ter zero ou mais dos seguintes conjunto de bits:

Valor param	Descrição param
`NOT_NULL_FLAG`	Campo não pode ser `NULL`
`PRI_KEY_FLAG`	Campo é parte de uma chave primária
`UNIQUE_KEY_FLAG`	Campo é parte de uma chave única
`MULTIPLE_KEY_FLAG`	Campo é parte de uma chave não única

UNSIGNED_FLAG	Campo tem o atributo UNSIGNED
ZEROFILL_FLAG	Campo tem o atributo ZEROFILL
BINARY_FLAG	Campo tem o atributo BINARY
AUTO_INCREMENT_FLAG	Campo tem o atributo AUTO_INCREMENT
ENUM_FLAG	Campo é um ENUM (obsoleto)
SET_FLAG	Campo é um SET (obsoleto)
BLOB_FLAG	Campo é um BLOB ou TEXT (obsoleto)
TIMESTAMP_FLAG	Campo é um TIMESTAMP (obsoleto)

Uso dos parâmetros BLOB_FLAG, ENUM_FLAG, SET_FLAG, e TIMESTAMP_FLAG foram obsoletos porque eles indicavam o tipo de um campo e não um atributo do tipo. É preferível testar campo->tipo para FIELD_TYPE_BLOB, FIELD_TYPE_ENUM, FIELD_TYPE_SET, ou FIELD_TYPE_TIMESTAMP.

O seguinte exemplo ilustra o uso típico do valor param:

```
if (campo->param & NOT_NULL_FLAG)
    printf("Campo não pode ser nulo\n");
```

Você pode usar as seguintes macros para determinar o status dos valores param:

Status param	Descrição
IS_NOT_NULL(param)	Verdadeiro se se este campo é definido como NOT NULL
IS_PRI_KEY(param)	Verdadeiro de este campo é uma chave primária
IS_BLOB(param)	Verdadeiro se este campo é um BLOB ou TEXT (obsoleto; teste campo->tipo)

`unsigned int decimals`

O número de decimais para um campo numérico.

12.1.2 Visão Geral das Função da API C

As funções disponíveis na API C são resumidas aqui e descritas em maiores detalhes em uma seção posterior. Veja Seção 12.1.3 [Funções API C], Página 783.

Função	Descrição
mysql_affected_rows()	Retorna o número de linhas alteradas/deletadas/insweridas pela última consulta \ UPDATE, DELETE, ou INSERT.
mysql_change_user()	Muda o usuario em um banco de dados em uma conexão aberta.
mysql_character_set_name()	Retorna o nome do conjunto de carcters padrão para a conexão.
mysql_close()	Fecha ua conexão com o servidor
mysql_connect()	Se conecta ao servidro MySQL. Esta função está deprecad; utilize mysql_real_connect().
mysql_create_db()	Cria um banco de dados. Esta função está obsoleta; utiliza o comando SQL CREATE DATABASE.

mysql_data_seek()	Busca por uma número de linha arbitrário em um conjunto de resultados de uma consulta.
mysql_debug()	Faz um `DBUG_PUSH` com a string dada.
mysql_drop_db()	Apaga um banco de dados; Esta função esta obsoleta; utiliza o comando SQL `DROP DATABASE`.
mysql_dump_debug_info()	Faz o servidor escrever informações de depouração no log.
mysql_eof()	Determina quando a ulitma linha de um conjunto de resultados foi lida. Esta função foi obsoleta; Utilize `mysql_errno()` ou `mysql_error()`
mysql_errno()	Retorna o número de erro para a função MySQL chamada mais recentemente.
mysql_error()	Retorna a mensagem de erro para função MySQL chamada mais recentemente.
mysql_escape_string()	Escapa caracteres especiais em uma string para ser usada em uma instrução SQL.
mysql_fetch_field()	Retorna o tipo do próximo campo na tabela.
mysql_fetch_field_direct()	Retorna o tipo de um campo da tabela, dado um número do campo.
mysql_fetch_fields()	Retorna um vetor de todas as estruturas do campo.
mysql_fetch_lengths()	Retorna o tamanho de todas as colunas na linha atual.
mysql_fetch_row()	Busca o próximo registro no conjunto de resultados.
mysql_field_seek()	Coloca o cursor da coluna em uma coluna específica.
mysql_field_count()	Retorna o número de colunas resultantes da consulta mais recente.
mysql_field_tell()	Retorna a posição do cursos de campos usado pelo último `mysql_fetch_field()`.
mysql_free_result()	Libera a memória usada por um conjunto de resultados.
mysql_get_client_info()	Retorna a versão do cliente como uma string.
mysql_get_client_version()	Returna a versão do cliente como um inteiro.
mysql_get_host_info()	Retorna uma string descrevendo a conexão.
mysql_get_server_version()	Retorna o número da versão do servidor como um inteiro (Novo na versão 4.1)
mysql_get_proto_info()	Retorna a versão do protovolo usado para a conexão.

mysql_get_server_info()	Retorna o número da versão do servidor.
mysql_info()	Retorna informação sobre a consulta executada mais recentemente.
mysql_init()	Obtem ou inicializa uma estrutura `MYSQL`.
mysql_insert_id()	Retorna o ID gerado para uma coluna `AUTO_INCREMENT` pela consulta anterior.
mysql_kill()	Mata uma thread dada.
mysql_list_dbs()	Retorna o nome do banco de dados correspondente a uma expressão regular.
mysql_list_fields()	retorna nome de campos coincidindo com uma expressão regular.
mysql_list_processes()	Retorna uma lista das threads atuais do servidor.
mysql_list_tables()	Retorna os nomes de tabelas correspondente a uma expressão regular.
mysql_num_fields()	Retorna o número de coluans em um conjunto de resultados.
mysql_num_rows()	Retorna o número de linhas em um conjunto de resultados.
mysql_options()	Define opções de conexão para `mysql_connect()`.
mysql_ping()	Verifica se a conexão ao servidor está funcionando, reconectando se necessário.
mysql_query()	Executa uma consulta SQL especificada com uma string terminada com null.
mysql_real_connect()	Conecta ao servidor MySQL.
mysql_real_escape_string()	Escapa caracteres especiais em uma string para ser utilizada em uma instrução SQL, olhando na conta o conjunto de caracteres atual da conexão
mysql_real_query()	Executa uma consulta SQL especificada como uma string fixa.
mysql_reload()	Diz ao servidor pra recarregar a tabela de permissões
mysql_row_seek()	Busca por um offset de linha no resultado, usando o valor retornado de `mysql_row_tell()`.
mysql_row_tell()	Retorna a posição dio cursor de linhas.
mysql_select_db()	Seleciona um banco de dados.

mysql_set_server_option()	Define uma opção para a conexão (como `multi-statements`).
mysql_sqlstate()	Retorna o código de erro SQLSTATE para o último erro.
mysql_shutdown()	Desliga o servidor de banco de dados.
mysql_stat()	Retorna o status do servidor como uma string.
mysql_store_result()	Recupera um resultado completo para o cliente.
mysql_thread_id()	Retorna a identificação da thread atual.
mysql_thread_safe()	Retorna 1 se o cliente foi compilado como thread-safe.
mysql_use_result()	Inicia uma resultado recuperado registro por registro.
mysql_warning_count()	Retorna a contagem do aviso da instrução SQL anterior.
mysql_commit()	Faz um commits na transação (novo na versão 4.1).
mysql_rollback()	Faz um roll back na transação (novo na versão 4.1).
mysql_autocommit()	Muda o modo autocommit em ligado/desligado (novo na versão 4.1).
mysql_more_results()	Verifica se não existem mais resultados (novo na versão 4.1).
mysql_next_result()	Retorna/Inicia o próximo resultado em execuções consultas múltiplas (inovo na versão 4.1).

Para se conectar ao servidor, chame `mysql_init()` para iniciar um manipulador de conexão, então chame `mysql_real_connect()` com este manipulador (com informações de nome de máquina, usuários e senha). Conectado, `mysql_real_connect()` define o parâmetro `reconnect` (parte da estrutura MYSQL) para um valor de `1`. Este parâmetro indica, no evento que uma consulta não pode ser realizada por perda de conexão, para tentar reconectar ao servidor ao antes de desistir. Quando não precisar mais da conexão, chame `mysql_close()` para terminá-la.

Enquanto a conexão estiver ativa, o cliente pode enviar consultas SQL para o servidor usando `mysql_query()` ou `mysql_real_query()`. A diferença entre os dois é que `mysql_query()` espera que a consulta seja especificada como uma string terminada em null, enquanto `mysql_real_query()` espera um string de tamanho fixa. Se a string conter dados binários (a qual pode incluir bytes null), vocêdeve usar `mysql_real_query()`.

Para cada consulta não-`SELECT` (por exemplo, `INSERT`, `UPDATE`, `DELETE`), você pode descobrir quantas linhas foram alteradas (afetadas) chamando `mysql_affected_rows()`.

Para consultas `SELECT`, você retorna os registros selecionados como um resultado. (Note que algumas intruções são como a `SELECT` ao retornar registros. Elas incluem `SHOW`, `DESCRIBE` e `EXPLAIN`. elas devem ser tratadas da mesma maneira que instruções `SELECT`.)

Existem dois modos para um cliente processa o resultado. Um mode é recuperar todo o resultado de uma vez chamando `mysql_store_result()`. Esta função busca no servidor todas as linhas retornadas pela consulta e as armazena no cliente. O segundo modo é o

cliente iniciar um retorno do resultado registro por registro chamando `mysql_use_result()`. Esta função inicia o retorno, mas não busca realmente nenhuma linha do servidor.

Em ambos os casos, acesse registros chamando `mysql_fetch_row()`. Com `mysql_store_result()`, `mysql_fetch_row()` acessa registros que já tenham sido buscado do servidor. Com `mysql_use_result()`, `mysql_fetch_row()` recupera, na verdade, o registro do servidor. Informações sobre o tamanho dos dados em cada registro é disponível pela chamada `mysql_fetch_lengths()`.

Depois de finalizar o uso do resultado, chame `mysql_free_result()` para liberar a memória usada por ele.

Os dois mecanismos de recuperação são complementares. Programas clientes devem escolher a abordagem mais apropriada para suas necessidades. Na prática, clientes tendem a utilizar `mysql_store_result()`.

Uma vantagem de `mysql_store_result()` é que pelo fato de todos os registros serem trazidos para o cliente, você não só pode acessar registros sequencialmente, mas também pode mover para tarz e para frente no resultado utilizando `mysql_data_seek()` ou `mysql_row_seek()` para altera a posição atual do registro no resultado. Você também pode saber quantas linhas existem chamando `mysql_num_rows()`. Por outro lado, a necessidade de memória para `mysql_store_result()` pode ser muito alta para resultados grandes e você encontrará como mais facilidade condições de estouro de memória.

Uma vantagem de `mysql_use_result()` é que o clientes exige menos memória para o resultado porque ele mantem apenas um registro por vez (por haver menor sobrecarga de alocação, `mysql_use_result()` pode ser mais rápido). As desvantagens são que você deve processar cada registro rapidamente para evitar prender o servidor, você não tem acesso aleatório aos registros no resultado (você só pode acessá-los sequencialmente) e você não sabe quantos registros existem no resultado até que você recupere todos eles. Além disso, você **deve** recuperar todos os registros mesmo que você já tenham encontrado a informação que procura antes do finalizar o conjunto de resultados.

A API torna possível para os clientes responder apropriadamente as consultas (recuperando somente os regiostros necessários) sem saber se a consulta é uma instrução `SELECT` ou não. Você pode fazer isto chamando `mysql_store_result()` depois de cada `mysql_query()` (ou `mysql_real_query()`). Se o resultado for obtido com sucesso, a consulta foi um `SELECT` e você pode ler os registros. Se a obtenção do resultado falhar, chame `mysql_field_count()` para determinar se o resultado era o esperado. Se `mysql_field_count()` retornar zero, a consulta não retornou nenhum dado (indicando que ela era um `INSERT`, `UPDATE`, `DELETE`, etc.), e não era esperado que retornasse registros. Se `mysql_field_count()` é diferente de zero, a consulta deveria retornar registros, mas não o fez. Isto indica que a consulta foi um `SELECT` que falhou. Veja a descrição de `mysql_field_count()` para um exemplo de como deve ser feito.

`mysql_store_result()` e `mysql_use_result()` permitem que você obtenha informação sobre os campos que montam o resultado (o número de campos, os seus nome e tipos, etc.) Você pode acessar informações de campo sequencialmente dentro dos registros chamando `mysql_fetch_field()` repetidamente, ou pelo número do campo dentro do registro chamando `mysql_fetch_field_direct()`. A posição atual do cursor de campos pode ser alterada cahamando `mysql_field_seek()`. Definir o cursor de campo afeta chamadas

subsequentes de `mysql_fetch_field()`. Você também pode conseguir informações de todos os campos de uma só vez chamando `mysql_fetch_fields()`.

Para detectar e relatar problemas, o MySQL fornace acesso a informações de erro através das funções `mysql_errno()` e `mysql_error()`. Elas retornam o código de erro ou a mensagem de erro para a função chamada mais recentemente que tenha tido sucesso ou que tenha falhado, permitindo a você determinar quando um erro ocorreu e qual foi ele.

12.1.3 Descrição das Funções da API C

Nas descrições a seguir, um parâmetro ou valor retornado `NULL` significa `NULL` no sentido da linguagem de programação C, não um valor `NULL` do MySQL.

Funções que retornam um valor geralmente retornam um ponteiro ou um inteiro. A menos que seja especificado, funcões que retornam um ponteiro, retornam um valor diferente de `NULL` para indicar sucesso ou um valor `NULL` para indicar um erro, e funções que retornam um inteiro, retoprnam zero para indicar sucesso ou um valor diferente de zero para indicar um erro. A menos que a descrição da função diga algo diferente, não faça teste com outro valor além do zero.

```
if (result)                  /* correct */
    ... error ...

if (result < 0)              /* incorrect */
    ... error ...

if (result == -1)            /* incorrect */
    ... error ...
```

Quando uma função retornar um erro, a subsecao **Erros** de descrição de funções lista os possíveis tipos de erro. Você pode descobrir quais deles ocorreu chamando `mysql_errno()`. Uma representação string do erro pode ser obtida chamando `mysql_error()`.

12.1.3.1 `mysql_affected_rows()`

`my_ulonglong mysql_affected_rows(MYSQL *mysql)`

Descrição

Retorna o número de registros alterados pelo último `UPDATE`, deletados elo último `DELETE` ou inseridos pelo último `INSERT`. Pode ser chamado imediatamente após `mysql_query()` para instruções `UPDATE`, `DELETE`, ou `INSERT`. Para instruções `SELECT`, `mysql_affected_rows()` funciona como `mysql_num_rows()`.

Valor Retornado

Um inteiro maior que zero indica o número de registros afetados ou recuperados. Zero indica que nenhum registro foi atualizado por uma instrução `UPDATE`, nenhuma linha foi encontrada pela cláusula `WHERE` na consulta ou a consulta ainda não foi executada. -1 indica que a

consulta retornou um erro ou que, para uma consulta `SELECT`, `mysql_affected_rows()` foi chamado antes da chamada `mysql_store_result()`.

Erros

Nenhum.

Exemplo

```
mysql_query(&mysql,"UPDATE products SET cost=cost*1.25 WHERE group=10");
printf("%ld products updated",(long) mysql_affected_rows(&mysql));
```

Se se for especificado o parâmetro `CLIENT_FOUND_ROWS` ao conectar no `mysqld`, `mysql_affected_rows()` retornará o número de linhas encontardos pela cláusula `WHERE` para a instrução `UPDATE`.

Note que quando for utilizado um comando `REPLACE`, `mysql_affected_rows()` retornará 2 se o novo registro substituir um mais antigo. Isto é porque neste caso um registro foi inserido e depois os registros duplicados foram deletados.

12.1.3.2 `mysql_change_user()`

```
my_bool mysql_change_user(MYSQL *mysql, const char *user, const char
*password, const char *db)
```

Descrição

Altera o usuário é faz com que o banco de dados especificado por `db` se torne o banco de dados padrão (atual) na conexão especificada por `mysql`. Em consultas subsequentes este banco de dados é o padrão para referências a tabelas que não especificam o banco de dados explicitamente.

Esta função foi introduzida na versão do MySQL.

`mysql_change_user()` falha a menos que o usuário conectado possa ser autenticado ou se ele não tiver permissão para utilizar o banco de dodos. Neste caso o usuário e o banco de dados não são alterados.

O parâmetro `db` pode ser definido como `NULL` se você não dseseja ter um banco de dados padrão.

A partir da versão 4.0.6 do MySQL este comando sempre fará `ROLLBACK` de qualquer transação ativa, fecha todas as tabelas temporárias, destrava todas as tabelas bloqueadas e volta a um estado como se tivesse feito uma inova conexão. Isto irá acontecer mesmo se o usuário não foi alterado.

Valor Retornado

Zero se obteve successo. Diferente de zero se ocorreu um erro.

Erros

O mesmo que pode ser obtido com `mysql_real_connect()`.

`CR_COMMANDS_OUT_OF_SYNC`
: Comandos forma executados em ordem inapropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL finalizou.

`CR_SERVER_LOST`
: A conexão ao servidor foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

`ER_UNKNOWN_COM_ERROR`
: O servidor MySQL não possui este comando (provavelmente um versão mais antiga)

`ER_ACCESS_DENIED_ERROR`
: O usuário ou a senha estavam errados.

`ER_BAD_DB_ERROR`
: O banco de dados não existe.

`ER_DBACCESS_DENIED_ERROR`
: O usuário não tem direitos de acessoa este banco de dados.

`ER_WRONG_DB_NAME`
: O nome de banco de dados é muito grande.

Exemplo

```
if (mysql_change_user(&mysql, "user", "password", "new_database"))
{
   fprintf(stderr, "Failed to change user.  Error: %s\n",
           mysql_error(&mysql));
}
```

12.1.3.3 `mysql_character_set_name()`

```
const char *mysql_character_set_name(MYSQL *mysql)
```

Descrição

Retorna o conjunto de caracteres padrão para a conexão atual.

Valor Retornado

O conjunto de carcteres padrão

Erros

Nenhum.

12.1.3.4 `mysql_close()`

```
void mysql_close(MYSQL *mysql)
```

Descrição

feca uma conexão aberta anteriormente. `mysql_close()` também desaloca o ponteiro do manipulador da conexão para o `mysql` se ele tiver sido alocado automaticamente por `mysql_init()` ou `mysql_connect()`.

Valor Retornado

Nenhum.

Erros

Nenhum.

12.1.3.5 `mysql_connect()`

```
MYSQL *mysql_connect(MYSQL *mysql, const char *host, const char *user, const
char *passwd)
```

Descrição

A função está obsoleta. É melhor utilizar `mysql_real_connect()`.

`mysql_connect()` tenta estabelecer uma conexão a um banco de dados MySQL executando em `host`. `mysql_connect()` deve completar com suceso antes que você podssa executar qualquer uma das função da API, com a exceção de `mysql_get_client_info()`.

O significado dos parâmetros são os mesmos que os parâmetros correspondentes para `mysql_real_connect()` com a diferença que o parâmetro de conexão pode ser `NULL`. Neste caso a API C aloca memória para a estrutura de conexão automaticamente e a libera quando você chamar `mysql_close()`. A disvantagem desta abordagem é que você não pode retornar uma mensagem de erro se a conexão falhar. (Para obter informações de erro de `mysql_errno()` ou `mysql_error()`, você deve fornecer um ponteiro `MYSQL` válido.)

Valor Retornado

O mesmo de `mysql_real_connect()`.

Erros

O mesmo de `mysql_real_connect()`.

12.1.3.6 `mysql_create_db()`

```
int mysql_create_db(MYSQL *mysql, const char *db)
```

Descrição

Cria o banco de dados nomeado pelo parâmetro `db`.

Esta função está obsoleta. É melhor utilizar `mysql_query()` para comandar uma instrução SQL `CREATE DATABASE`.

Valor Retornado

Zero se o banco de dados foi criado com sucesso. Diferente de zero se ocorreu um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

Exemplo

```
if(mysql_create_db(&mysql, "my_database"))
{
   fprintf(stderr, "Failed to create new database.  Error: %s\n",
           mysql_error(&mysql));
}
```

12.1.3.7 `mysql_data_seek()`

```
void mysql_data_seek(MYSQL_RES *result, my_ulonglong offset)
```

Descrição

Busca um registro arbitrário em um resultado de uma consulta. O valor do offset é um número de linha e deve estar em uma faixa de `0` até `mysql_num_rows(stmt)-1`.

Esta função exige que a estrutura do resultado contenha todo o resultado da consulta, assim `mysql_data_seek()` só pode ser usado em conjunto com `mysql_store_result()`, não com `mysql_use_result()`.

Valor Retornado

Nenhum.

Erros

Nenhum.

12.1.3.8 `mysql_debug()`

```
void mysql_debug(const char *debug)
```

Descrição

Faz um `DBUG_PUSH` com a string dada. `mysql_debug()` usa a biblioteca de depuração Fred Fish. Para utilizar esta função você deve compilar a biblioteca cliente para suportar depuração. Veja Seção E.1 [Depurando o servidor], Página 1079. Veja Seção E.2 [Depurando o cliente], Página 1085.

Valor Retornado

Nenhum.

Erros

Nenhum.

Exemplo

A chamada mostrada aqui faz com que a biblioteca cliente gere um arquivo de rastreamento '`/tmp/client.trace`' na máquina cliente:

```
mysql_debug("d:t:O,/tmp/client.trace");
```

12.1.3.9 `mysql_drop_db()`

```
int mysql_drop_db(MYSQL *mysql, const char *db)
```

Descrição

Apaga o banco de dados nomeado pelo parâmetro `db`.

Esta função está obsoleta. É melhor utilizar `mysql_query()` para realizar uma instrução SQL `DROP DATABASE`.

Valor Retornado

Zero se o banco de dados foi apagdo com sucesso. Diferente de zero ocorreu um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

Exemplo

```
if(mysql_drop_db(&mysql, "my_database"))
  fprintf(stderr, "Failed to drop the database: Error: %s\n",
          mysql_error(&mysql));
```

12.1.3.10 `mysql_dump_debug_info()`

```
int mysql_dump_debug_info(MYSQL *mysql)
```

Descrição

Instrui o servidor a gravar algumas informações de depuração no log. Para funcionar, o usuário conectado deve ter pivilégio `SUPER`.

Valor Retornado

Zero se o comando obteve sucesso. Diferete de zero se ocorreu um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.11 `mysql_eof()`

```
my_bool mysql_eof(MYSQL_RES *result)
```

Descrição

Esta função está obsoleta. `mysql_errno()` ou `mysql_error()` podem ser usados em seu lugar.

`mysql_eof()` determina se o último registro de um resultado foi lido.

Se você buscar um resultado com um chamada `mysql_store_result()` bem sucedida, o cliente recebe todo o resultado em uma operação. Neste caso, é um valor `NULL` retornado de `mysql_fetch_row()` sempre significa que o fim do resultado foi atingido e não é necessário chamar `mysql_eof()`. Quando usado com `mysql_store_result()`, `mysql_eof()` sempre retornará verdadeiro.

Por outro lado, se você utilizar `mysql_use_result()` para iniciar um resultado recuperado, as linhas do conjunto são obtido do servidor uma a uma, chamando `mysql_fetch_row()` repetidamente. Como pode ocorrer um erro na conexão durante este processo, um valor `NULL` retornado de `mysql_fetch_row()` não significa, necessáriaemente, que o fim do resultado fo atingido normalmente. Neste caso, você pode utilizar `mysql_eof()` para determinar o que aconteceu. `mysql_eof()` retorna um valor diferente de zero se o fim do resultaod foi atingido e zero se ocorreu um erro.

Historicamente, `mysql_eof()` é preterido pelas funções de erro padrão do MySQL `mysql_errno()` e `mysql_error()`. Como estas funções de erro fornecem a mesma informação, o uso das duas últimas é preferido sobre `mysql_eof()`, a qual está obsoleta. (De fato, elas fornecem mais informações, porque `mysql_eof()` retorna apenas um valor booleano enquanto as funções de erro indicam uma razão para a ocorrência do erro quando ele ocorre).

Valor Retornado

Zero se nenhum erro ocorreu. Diferente de zero o fim do resultado foi atingido.

Erros

Nenhum.

Exemplo

Os exemplos seguintes mostram como você deve usar `mysql_eof()`:

```
mysql_query(&mysql,"SELECT * FROM some_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
{
    // faz algo com os dados
}
if(!mysql_eof(result))  // mysql_fetch_row() falha devido a um erro
{
    fprintf(stderr, "Error: %s\n", mysql_error(&mysql));
}
```

No entanto, você pode conseguir o mesmo efeito com as funções de erro padrões do MySQL:

```
mysql_query(&mysql,"SELECT * FROM some_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
{
    // faz algo com os dados
}
if(mysql_errno(&mysql))  // mysql_fetch_row() falha devido a um erro
{
    fprintf(stderr, "Error: %s\n", mysql_error(&mysql));
}
```

12.1.3.12 `mysql_errno()`

`unsigned int mysql_errno(MYSQL *mysql)`

Descrição

Para a conexão especificada pelo `mysql`, `mysql_errno()` retorna o código de erro para a função API chamada mais recentemente que tenha obtido sucesso ou falhado. Um valor de retorno de zero significa que um erro ocorreu. Números de mensagens de erro de clientes são listados no arquivo de cabeçalho '`errmsg.h`' do MySQL. Números de mensagem de erros do servidor são listados no arquivo '`mysqld_error.h`'. Na distribuição fonte do MySQL você pode encontrar uma lista completa de núeros de mensagens de erro no arquivo '`Docs/mysqld_error.txt`'. Os códigos de erros do servidor estão listados em Seção 13.1 [Error-returns], Página 889.

Note que algumas funções como `mysql_fetch_row()` não configuram o `mysql_errno()` se elas obterem sucesso.

Uma regra do dedão é que todas as funções que precisam perguntar ao servidor por informação irão zerar `mysql_errno()` se obterem sucesso.

Valor Retornado

Um valor de código de erro para a última chamada mysql_xxx, se ele falhar, Zero significa que nenhum erro ocorreu.

Erros

Nenhum.

12.1.3.13 `mysql_error()`

```
const char *mysql_error(MYSQL *mysql)
```

Descrição

Para a conexão especificada por `mysql`, `mysql_error()` retorna um string terminada em null contendo a mensagem de erro para a função de API chamda mais recentemente que tenha falhado. Se a função não falhou, o valor de retorno de `mysql_error()` pode ser o erro anterior ou uma string vazia para indicar que não ocorreu erro.

Uma regra do dedão é que todas as funções que precisam pedir informação ao servidor irão zerar `mysql_error()` se obterem sucesso.

Para todas as funções que zeram `mysql_errno`, os seguintes dois testes são equivalentes:

```
if(mysql_errno(&mysql))
{
    // ocorreu um erro
}

if(mysql_error(&mysql)[0] != '\0')
{
    // ocorreu um erro
}
```

A língua da mensagem de erro do cliente pode ser alterada recompilando a biblioteca do cliente MySQL. Atualmente você pode escolher mensagens de erro em várias línguas diferentes. Veja Seção 4.7.2 [Languages], Página 327.

Valor Retornado

Uma string terminada em null que descreve um erro. Uma string vazia se nenhum erro ocorrer.

Erros

Nenhum.

12.1.3.14 mysql_escape_string()

Você deve usar `mysql_real_escape_string()` em seu lugar!

Esta função é identica a `mysql_real_escape_string()` exceto que `mysql_real_escape_string()` pega um manipulador de cnexão como seu primeiro argumento e escapa a string de acordo com a conjunto de caracteres padrão. `mysql_escape_string()` não utiliza um argumento de conexão e não respeita o conjunto de caracteres atual.

12.1.3.15 mysql_fetch_field()

```
MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *result)
```

Descrição

Retorna a definição de uma coluna de um resultado como uma estrutura `MYSQL_FIELD`. Chame esta função repetidamente para retornar informações sobre todas as colunas no resultado. `mysql_fetch_field()` retorna `NULL` quando não existirem mais campos.

`mysql_fetch_field()` é definido para retornar a informação do primeiro campo cada vez que você executar uma nova consulta `SELECT`. O campo retornado por `mysql_fetch_field()` também é afetado pela chamadas `mysql_field_seek()`.

Se vovê tiver chamado `mysql_query()` para realizar um `SELECT` em uma tabela mas não tiver chamado `mysql_store_result()`, MySQL retorna o tamanho padrão do blob (8K bytes) quando chamar `mysql_fetch_field()` para saber o tamanho de um campo `BLOB`. (O tamanho de 8 k é escolhido porque o MySQL não sabe o tamanho máximo do `BLOB`. Ele pode ser configurado algumas vezes.) Uma vez retornado o resultado, `campo->tamanho_max` contém o tamanho da maior valor para esta coluna em uma consulta específica.

Valor Retornado

A estrutura `MYSQL_FIELD` para a coluna atual. `NULL` não houver mais colunas.

Erros

Nenhum.

Exemplo

```
MYSQL_FIELD *field;

while((field = mysql_fetch_field(result)))
{
    printf("field name %s\n", field->name);
}
```

12.1.3.16 mysql_fetch_fields()

```
MYSQL_FIELD *mysql_fetch_fields(MYSQL_RES *result)
```

Descrição

Retorna um vetor de todas as estruturas `MYSQL_FIELD` no resultado. Cada estrutura fornece a definição do campo para uma coluna do resultado.

Valor Retornado

Um vetor da estrutura `MYSQL_FIELD` para todas as colunas no resultado.

Erros

Nenhum.

Exemplo

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *fields;

num_fields = mysql_num_fields(result);
fields = mysql_fetch_fields(result);
for(i = 0; i < num_fields; i++)
{
   printf("Field %u is %s\n", i, fields[i].name);
}
```

12.1.3.17 mysql_fetch_field_direct()

```
MYSQL_FIELD *mysql_fetch_field_direct(MYSQL_RES *result, unsigned int
fieldnr)
```

Descrição

Dado um número de campo `fieldnr` para uma colua em resultado, retorna a informação de campo daquela coluna como uma estrutura `MYSQL_FIELD` Você pode utilizar esta função para retornar a definição para uma coluna arbitrária. O valor de `fieldnr` deve estar na faixa de 0 a `mysql_num_fields(result)-1`.

Valor Retornado

A estrutura `MYSQL_FIELD` para uma coluna específica.

Erros

Nenhum.

Exemplo

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *field;

num_fields = mysql_num_fields(result);
for(i = 0; i < num_fields; i++)
{
    field = mysql_fetch_field_direct(result, i);
    printf("Field %u is %s\n", i, field->name);
}
```

12.1.3.18 mysql_fetch_lengths()

```
unsigned long *mysql_fetch_lengths(MYSQL_RES *result)
```

Descrição

Retorna o tamanho da coluna do registro atual em um resultado. Se você planeja copiar calores dos compos, esta informação de tamanho é útil também para a otimização, porque você pode evitar a chamada `strlen()`. Se o resultado contém dados biários, você **deve**utilizar esta função para determinar o tamanho dos dados, pois `strlen()` retorna um valor incorreto para quaquer campo contendo caracteres nulos.

O tamanho para colunas vazias e para colunas contendo valores `NULL` é zero. Para ver como distnguir este dois casos, veja a descrição de `mysql_fetch_row()`.

Valor Retornado

Um vetor de unsigned long integers (inteiros longos sem sinal) representando o tamanho de cada coluna (não incluindo nenhuma caracter nulo). `NULL` se ocorrer um erro.

Erros

`mysql_fetch_lengths()` só é válido para o registro atual no resultado. Ele retorna `NULL` se você chamá-lo antes de `mysql_fetch_row()` ou depois de retornar todos os registros em um resultado.

Exemplo

```
MYSQL_ROW row;
```

```
unsigned long *lengths;
unsigned int num_fields;
unsigned int i;

row = mysql_fetch_row(result);
if (row)
{
    num_fields = mysql_num_fields(result);
    lengths = mysql_fetch_lengths(result);
    for(i = 0; i < num_fields; i++)
    {
         printf("Column %u is %lu bytes in length.\n", i, lengths[i]);
    }
}
```

12.1.3.19 mysql_fetch_row()

`MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)`

Descrição

Recuera o próximo registro do resultado. Quando usado depois de `mysql_store_result()`, `mysql_fetch_row()` retorna `NULL` quando não houver mais registros para retornar. Quando usado depois de `mysql_use_result()`, `mysql_fetch_row()` retorna `NULL` quando não houver mais registros para retornar ou ocorrer um erro.

O número de valores no registro é dado por `mysql_num_fields(result)`. Se `row` guarda o valor retornado de uma chamada `mysql_fetch_row()`, apontadores para os valores são acessados como `row[0]` a `row[mysql_num_fields(result)-1]`. Valores `NULL` no registro são indicados por apontadores `NULL`.

Os tamanhos dos valores do campo no registro poden ser obtidos chamando `mysql_fetch_lengths()`. Campos vazios e campos contendo `NULL` tem tamanho 0; você pode distingui-los verificando o apontador para o valor do campo. Se o apontador é `NULL`, o campo é `NULL`; senão o campo está vazio.

Valor Retornado

Uma estrutura `MYSQL_ROW` para o próximo registro. `NULL` se não houver mais linhas para retornar ou ocorrer um erro.

Erros

Note que o erro não é zerado entre as chamadas a `mysql_fetch_row()`

`CR_SERVER_LOST`
: A conexão com o servidor foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

Exemplo

```
MYSQL_ROW row;
unsigned int num_fields;
unsigned int i;

num_fields = mysql_num_fields(result);
while ((row = mysql_fetch_row(result)))
{
   unsigned long *lengths;
   lengths = mysql_fetch_lengths(result);
   for(i = 0; i < num_fields; i++)
   {
       printf("[%.*s] ", (int) lengths[i], row[i] ? row[i] : "NULL");
   }
   printf("\n");
}
```

12.1.3.20 `mysql_field_count()`

`unsigned int mysql_field_count(MYSQL *mysql)`

Se você estiver utilizando uma versão anterior a versão 3.22.24 do MySQL, você deve utilizar `unsigned int mysql_num_fields(MYSQL *mysql)`.

Descrição

Retorna o número de colunas para a consulta mais recente na conexão.

Normalmente esta função é utilizada quando `mysql_store_result()` retorna `NULL` (então você não possui um apontador para o resultado). Neste caso, você pode chamar `mysql_field_count()` para determinar se `mysql_store_result()` não produziu um resultado vazio. Isto permite que o programa cliente tome a ação aprpriada sem saber se a consulta foi uma instrução `SELECT` (ou do mesmo tipo). O exemplo mostrado aqui ilustra como isto pode ser feito.

Veja Seção 12.1.12.1 [`NULL mysql_store_result()`], Página 861.

Valor Retornado

Um unsigned integer (inteiro sem sinal) representando o número de campo em um resultado.

Erros

Nenhum.

Exemplo

```
MYSQL_RES *result;
unsigned int num_fields;
unsigned int num_rows;

if (mysql_query(&mysql,query_string))
{
    // error
}
else // query succeeded, process any data returned by it
{
    result = mysql_store_result(&mysql);
    if (result)  // there are rows
    {
        num_fields = mysql_num_fields(result);
        // retrieve rows, then call mysql_free_result(result)
    }
    else  // mysql_store_result() returned nothing; should it have?
    {
        if(mysql_field_count(&mysql) == 0)
        {
            // query does not return data
            // (it was not a SELECT)
            num_rows = mysql_affected_rows(&mysql);
        }
        else // mysql_store_result() should have returned data
        {
            fprintf(stderr, "Error: %s\n", mysql_error(&mysql));
        }
    }
}
```

Uma alternativa é substituir a chamada `mysql_field_count(&mysql)` com `mysql_errno(&mysql)`. Neste caso, você está verificando diretamente um erro de `mysql_store_result()` em vez de conferir o valor de `mysql_field_count()` se a instrução foi uma `SELECT`.

12.1.3.21 `mysql_field_seek()`

`MYSQL_FIELD_OFFSET mysql_field_seek(MYSQL_RES *result, MYSQL_FIELD_OFFSET offset)`

Descrição

Define o cursor campo com o offset dado. A próxima chamada para `mysql_fetch_field()` irá recuperar a definição de campo da coluna associada com o offset.

Para buscar o inicio de um registro, passe zero como valor do `offset`.

Valor Retornado

O valor anterior do cursor de campo.

Erros

Nenhum.

12.1.3.22 `mysql_field_tell()`

```
MYSQL_FIELD_OFFSET mysql_field_tell(MYSQL_RES *result)
```

Descrição

Retorna a posição do cursos do campo usado pelo último `mysql_fetch_field()`. Este valor pode ser usado como um argumento para `mysql_field_seek()`.

Valor Retornado

O offset atual do cursor de campo.

Erros

Nenhum.

12.1.3.23 `mysql_free_result()`

```
void mysql_free_result(MYSQL_RES *result)
```

Descrição

Libera a memória alocada para o resultado por `mysql_store_result()`, `mysql_use_result()`, `mysql_list_dbs()`, etc. Quando você finalizar o uso do resultado, você deve liberar a memória utilizada chamando `mysql_free_result()`.

Valor Retornado

Nenhum.

Erros

Nenhum.

12.1.3.24 mysql_get_client_info()

```
char *mysql_get_client_info(void)
```

Descrição

Retorna uam string que representa a versão da biblioteca cliente.

Valor Retornado

Uma string representando a versão da biblioteca cliente do MySQL.

Erros

Nenhum.

12.1.3.25 mysql_get_client_version()

```
unsigned long mysql_get_client_version(void)
```

Descrição

Retorna um inteiro que representa a versão da biblioteca cliente. O valor tem o formato `XYYZZ` onde `X` é a versão principal, `YY` é o nivel da distribuicão e `ZZ` é o número da versão dentro do nível da distribuição. Por exemplo, um valor de `40102` representa uma biblioteca cliente na versão `4.1.2`.

Valores de Retorno

Um inteiro que representa a versão da biblioteca clientes do mysql.

Erros

Nenhum.

12.1.3.26 mysql_get_host_info()

```
char *mysql_get_host_info(MYSQL *mysql)
```

Descrição

Retorna uma string descrevendo o tipo da conexão em uso, incluindo o nome da maquina servidora.

Valor Retornado

Uma string respresntando o nome da máquina servidora e o tipo de conexão.

Erros

Nenhum.

12.1.3.27 mysql_get_proto_info()

```
unsigned int mysql_get_proto_info(MYSQL *mysql)
```

Descrição

Retorna a versão do protocolo usado pela conexão atual.

Valor Retornado

Um unsigned integer (inteiro sem sinal) representando a versão do protocolo usado pela conexão atual.

Erros

Nenhum.

12.1.3.28 mysql_get_server_info()

```
char *mysql_get_server_info(MYSQL *mysql)
```

Descrição

Retorna um string que representa o número da versão do servidor.

Valor Retornado

Um string representando o número da versão do servidor.

Erros

Nenhum.

12.1.3.29 mysql_get_server_version()

```
unsigned long mysql_get_server_version(MYSQL *mysql)
```

Descrição

Retorna o número de versão do servidor como um inteiro (novo na versão 4.1)

Valor Retornado

Um número que representa a versão do servidor MySQL no formato:

versão_principal*10000 + versão_menor*100 + sub_versão

Por exemplo, 4.1.0 é retornado como 40100.

Ela é útil para determinar a versão do servidor rapidamente em um programa cliente para saber se algumas capacidades existem.

Erros

Nenhum.

12.1.3.30 `mysql_info()`

```
char *mysql_info(MYSQL *mysql)
```

Descrição

Retorna um string fornecendo informação sobre a consulta executada mais recentemente, mas apenas para as instruções listadas aqui. Para outras inastruções, `mysql_info()` retorna `NULL`. O formato da string varia dependendo do tipo de consulta, como descrito aqui. Os números são apenas ilustrativos; a string irá conter valores apropriados para a consulta.

`INSERT INTO ... SELECT ...`
: Formato da string: `Records: 100 Duplicates: 0 Warnings: 0`

`INSERT INTO ... VALUES (...),(...),(...)...`
: Formato da string: `Records: 3 Duplicates: 0 Warnings: 0`

`LOAD DATA INFILE ...`
: Formato da string: `Records: 1 Deleted: 0 Skipped: 0 Warnings: 0`

`ALTER TABLE`
: Formato da string: `Records: 3 Duplicates: 0 Warnings: 0`

`UPDATE` Formato da string: `Rows matched: 40 Changed: 40 Warnings: 0`

Note que `mysql_info()` retorna um valor não-`NULL` para `INSERT ... VALUES` somente na forma de múltiplas linhas da instrução (isto é, apenas se uma lista com vários valores é especificada).

Valor Retornado

Uma string represntando informação adicional sobre a consulta executada mais recentemente. `NULL` se não houver nenhuma informação disponível para a consulta.

Erros

Nenhum.

12.1.3.31 mysql_init()

```
MYSQL *mysql_init(MYSQL *mysql)
```

Descrição

Aloca ou inicializa um objeto `MYSQL` apropriado para `mysql_real_connect()`. Se `mysql` é um ponteiro `NULL`, a função aloca, inicializa e retorna um novo objeto. Senão o objeto é inicializado e o endereço do objeto é retornado. Se `mysql_init()` aloca um novo objeto, ele será liberado quando `mysql_close()` for chamado para fechar a conexão.

Valor Retornado

Um handle `MYSQL*` inicializado. `NULL` se não houver memória suficiente para alocar o novo objeto.

Erros

Em caso de memória insuficiente, `NULL` é retornado.

12.1.3.32 mysql_insert_id()

```
my_ulonglong mysql_insert_id(MYSQL *mysql)
```

Descrição

Retorna o ID gerado para uma coluna `AUTO_INCREMENT` pela consulta anterior. Use esta função depois de ter realizado um consulta `INSERT` em uma tabela que contenha um campo `AUTO_INCREMENT`.

Note que `mysql_insert_id()` retorna `0` se a consulta anterior não gerar um valor `AUTO_INCREMENT`. Se você desejar salvar o valor para uso posterior, chame `mysql_insert_id()` imediatamente depois da consulta que gerou o valor.

Se a consulta anterior retornar um erro, o valor de `mysql_insert_id()` é indefinido.

`mysql_insert_id()` é atualizado depois de instruções `INSERT` e `UPDATE` que geram um valor `AUTO_INCREMENT` ou que definem um valor de coluna com `LAST_INSERT_ID(expr)`. Veja Seção 6.3.6.2 [Funções diversas], Página 547.

Note também que o valor da função SQL `LAST_INSERT_ID()` sempre contém o o valor `AUTO_INCREMENT` gerado mais recentemente e não é zerado entre as consultas porque o valor desta função é mantido no servidor.

Valor Retornado

O valor do campo `AUTO_INCREMENT` que foi atualizado pela consulta anterior. Retorna zero se não houve consultas anteriores na conexão ou se a consulta não atualizou o valor `AUTO_INCREMENT`.

Erros

Nenhum.

12.1.3.33 `mysql_kill()`

```
int mysql_kill(MYSQL *mysql, unsigned long pid)
```

Descrição

Diz para o servidor matar um thread especificada pelo `pid`.

Valor Retornado

Zero em caso de sucesso. Diferente de zero se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.34 `mysql_list_dbs()`

```
MYSQL_RES *mysql_list_dbs(MYSQL *mysql, const char *wild)
```

Descrição

Retorna um resultado com nome de banco de dados no servidor que correspondem a uma expressão regular especificada pelo parâmetro `wild`. `wild` pode conter o meta caracteres '`%`' ou '`_`', ou pode ser um ponteiro `NULL` para coreesponder a todos os banco de dados. Chamar `mysql_list_dbs()` é o mesmo que executar a consulta `SHOW databases [LIKE wild]`.

Você deve liberar o resultado com `mysql_free_result()`.

Valor Retornado

Um conjunto de resultados `MYSQL_RES` no caso de sucesso. `NULL` se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.35 `mysql_list_fields()`

```
MYSQL_RES *mysql_list_fields(MYSQL *mysql, const char *table, const char
*wild)
```

Descrição

Retorna um resultado contendo nomes de campos de uma tabela dada que correspondam a expressão regular especificada pelo parâmetro `wild`. `wild` pode conter os metacaracteres '`%`' ou '`_`', ou pode ser um ponteiro `NULL` para corresponder a todos os campos. Chamar `mysql_list_fields()` é o mesmo que executar a consulta `SHOW COLUMNS FROM nome_tabela [LIKE wild]`.

Note que é recomendado que você use `SHOW COLUMNS FROM nome_tabela` em vez de `mysql_list_fields()`.

Você deve liberar o resultado com `mysql_free_result()`.

Valor Retornado

Um conjunto de resultados `MYSQL_RES` em caso de sucesso. `NULL` se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.36 `mysql_list_processes()`

```
MYSQL_RES *mysql_list_processes(MYSQL *mysql)
```

Descrição

Retorna um resultado descrevendo a thread atual do servidor. É o mesmo tipo de informação relatado por `mysqladmin processlist` ou uma consulta `SHOW PROCESSLIST`.

Você deve liberar o resultado com `mysql_free_result()`.

Valor Retornado

Um conjunto de resultados `MYSQL_RES` em caso de sucesso. `NULL` se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.37 `mysql_list_tables()`

```
MYSQL_RES *mysql_list_tables(MYSQL *mysql, const char *wild)
```

Descrição

Retorna um resultado contendo nomes de tabelas no banco de dados atual que correspondam a expressão regular especificada pelo parâmetro `wild`. `wild` pode conter os mets caracteres '`%`' or '`_`', ou pode ser uma ponteiro `NULL` para corresponde a todas as tabelas. Chamar `mysql_list_tables()` é o mesmo que executar a consulta `SHOW tables [LIKE wild]`.

Você deve liberar o resultado com `mysql_free_result()`.

Valor Retornado

Um conjunto de resultados `MYSQL_RES` em caso de sucesso. `NULL` se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.38 `mysql_num_fields()`

```
unsigned int mysql_num_fields(MYSQL_RES *result)
```

ou

```
unsigned int mysql_num_fields(MYSQL *mysql)
```

A segunda forma não funciona na versão 3.22.24 ou mais novas do MySQL. Para passar um argumento `MYSQL*` você de utilizar `unsigned int mysql_field_count(MYSQL *mysql)` em seu lugar.

Descrição

Retorna o número de colunas em um resultado.

Note que você pode obter o número de colunas com um ponteiro para o conjunto de resultados ou para um manipulador (handle) de conexão. Você usaria o manipular de conexão se `mysql_store_result()` ou `mysql_use_result()` retorna `NULL` (então você não tem um ponteiro para o resultado). Neste caso, você pode chamar `mysql_field_count()` para determinar se `mysql_store_result()` não produziu um resultado vazio. Isto permite que o programa cliente tome a ação apropriada sem saber se a consulta foi uma instrução `SELECT` (ou do tipo `SELECT`). O exemplo mostrado abaixo ilustra como isto pode ser feito.

Veja Seção 12.1.12.1 [`NULL mysql_store_result()`], Página 861.

Valor Retornado

Um unsigned integer (inteiro sem sinal) representando o número de campos no conjunto de resultasdos.

Erros

Nenhum.

Exemplo

```
MYSQL_RES *result;
unsigned int num_fields;
unsigned int num_rows;

if (mysql_query(&mysql,query_string))
{
    // erro
}
else // query succeeded, process any data returned by it
{
    result = mysql_store_result(&mysql);
    if (result)  // existem resgitros
    {
        num_fields = mysql_num_fields(result);
        // retorna registros e chama mysql_free_result(result)
    }
    else  // mysql_store_result() retorna vazio; era esperado?
    {
        if (mysql_errno(&mysql))
{
           fprintf(stderr, "Error: %s\n", mysql_error(&mysql));
}
        else if (mysql_field_count(&mysql) == 0)
        {
            // consulta não retora dados
            // (ela não era um SELECT)
            num_rows = mysql_affected_rows(&mysql);
        }
    }
}
```

Uma alternativa (se você souber qyue a sua consulta retornou um resultado) é substituir a chamada `mysql_errno(&mysql)` pela verificação de se `mysql_field_count(&mysql)` é = 0. Isto só acontece se alguma coisa der errado.

12.1.3.39 `mysql_num_rows()`

`my_ulonglong mysql_num_rows(MYSQL_RES *result)`

Descrição

Retorna o número de linhas em um resultado.

O uso de `mysql_num_rows()` depende de se você utiliza `mysql_store_result()` ou `mysql_use_result()` para retornar o resultado. Se você usa `mysql_store_result()`, `mysql_num_rows()` pode ser chamado imediatamente. Se você usa `mysql_use_result()`, `mysql_num_`

`rows()` não retornará o valor correto até que todas as linhas no resultado tenham sido recuperadas.

Valor Retornado

O número de linhas no resultado.

Erros

Nenhum.

12.1.3.40 `mysql_options()`

```
int mysql_options(MYSQL *mysql, enum mysql_option option, const char *arg)
```

Descrição

Pode ser usado para definir opções extras de conexão e afetar o comportamento de uma conexão. Esta função pode ser chamada várias vezes para definir diversas opções.

`mysql_options()` deve ser chamado depois de `mysql_init()` e antes de `mysql_connect()` ou `mysql_real_connect()`.

O argumento `option` é a opção que você que definir; o argumento `arg` é o valor para a opção. Se a opção é um inteiro, então `arg` deve apontar para o valor do inteiro.

Valores possíveis para as opções:

Opção	Tipo de argumento	Função
`MYSQL_OPT_CONNECT_TIMEOUT`	`unsigned int *`	Tempo limite de conexão em segundos.
`MYSQL_OPT_COMPRESS`	Não usado	Usa o protocolo cliente/servidor compactado.
`MYSQL_OPT_READ_TIMEOUT`	`unsigned int *`	Limite de tempo para a leitura do servidor (funciona atualmente apenas no Windows em conexões TCP/IP)
`MYSQL_OPT_WRITE_TIMEOUT`	`unsigned int *`	Limite de tempo para a escrita no servidor (funciona atualmente apenas no Windows em conexões TCP/IP)
`MYSQL_OPT_LOCAL_INFILE`	ponteiro para unsigned integer opcional	Se nenhum ponteiro for dado ou se apontar para um `unsigned int != 0` o comando `LOAD LOCAL INFILE` está habilitado.
`MYSQL_OPT_NAMED_PIPE`	Não usado	Usa named pipes para conectar ao servidor MySQL no NT.

MYSQL_INIT_COMMAND	char *	Comando para executar ao conectar ao servidor MySQL. Será automaticamente executado ao se reconectar.
MYSQL_READ_DEFAULT_FILE	char *	Lê opções do arquivo de opções definido no lugar de 'my.cnf'.
MYSQL_READ_DEFAULT_GROUP	char *	Lê opções do grupo indicado no arquivo 'my.cnf' ou no arquivo especificado com MYSQL_READ_DEFAULT_FILE.
MYSQL_OPT_PROTOCOL	unsigned int *	Tipo de protocolo usado. Deve ser um dos valores apresentados em mysql_protocol_type definido no 'mysql.h'.
MYSQL_SHARED_MEMORY_BASE_NAME	char*	Nome do objeto em meória para comunicação com o servidor. Deve ser o mesmo que a opção --shared-memory-base-name usada para o servidor mysqld no qual você quer se conectar.

Note que o grupo client é sempre lido se você utiliza MYSQL_READ_DEFAULT_FILE ou MYSQL_READ_DEFAULT_GROUP.

O grupo especificado no arquivo de opçõs pode conter as seguintes opções:

Opção	Descrição
connect-timeout	Tempo limite de conexão em segundos. No Linux este tempo limite também é utilizado para esperar pela primeira resposta do servidor
compress	Utiliza o protocolo cliente/servidor compactado.
database	Conecta a este banco de dados se nenhum banco de dados for especificado no comando de conexão.
debug	Opções de depuração.
disable-local-infile	Disabilita o uso de LOAD DATA LOCAL.
host	Nome de máquina padrão.
init-command	Comando para executar ao conectar ao servidor MySQL. Será executado automaticamente ao reconectar.
interactive-timeout	O mesmo que o especificado em CLIENT_INTERACTIVE para mysql_real_connect(). Veja Seção 12.1.3.43 [mysql_real_connect], Página 813.
local-infile[=(0|1)]	Se não houver argumento ou o argumento for diferente de 0 habilita o uso de LOAD DATA LOCAL.
max_allowed_packet	Tamanho máximo dos pacotes que o cliente pode ler do servidor.
password	Senha padrão.
pipe	Usa named pipes para conectar ao servidor MySQL no NT.

`protocol=(TCP | SOCKET | PIPE | MEMORY)`	Qual protocolo usar ao conectar no servidor (Novo na versão 4.1)
`port`	Número padrão da porta.
`return-found-rows`	Diz ao `mysql_info()` para retornar registros encontrados no lugar de registros atualizados ao usar `UPDATE`.
`shared-memory-base-name=name`	Nome da memória comprtilhada utilizada para conectar ao servidor (o padrão é "MySQL"). Novo na versão 4.1.
`socket`	Número padrão do socket.
`user`	Usuário padrão.

Note que `timeout` foi substituido por `connect-timeout`, mas `timeout` ainda funcionará por enquanto.

Para maiores informações sobre arquivos de opções, veja Seção 4.1.2 [Arquivo de opções], Página 216.

Valor Retornado

Zero em caso de sucesso. Diferente de zero se você utilizar uma opção desconhecida.

Exemplo

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_OPT_COMPRESS,0);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"odbc");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
    fprintf(stderr, "Failed to connect to database: Error: %s\n",
          mysql_error(&mysql));
}
```

O exemplo acima diz ao cliente para usar o protocolo cliente/servidor compactado e ler a opção adicional da seção `odbc` no arquivo de opções '`my.cnf`'.

12.1.3.41 `mysql_ping()`

`int mysql_ping(MYSQL *mysql)`

Descrição

Verifica se a conexão ao servidor está funcionando. Se ela tiver caído é feita uma tentativa de conexão automaticamente.

Esta função pode ser usada pelos clientes que se ficam inativo por um longo tempo para verificar se o servidor fechou a conexão e reconectar se necessário.

Valor Retornado

Zero se o servidor estiver funcionando. Diferente de zero se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.42 `mysql_query()`

```
int mysql_query(MYSQL *mysql, const char *query)
```

Descrição

Executa uma consulta SQL apontada pela string terminada em null `query`. A consulta deve deve consistir de uma única instrução SQL. Você não deve adicionar ponto e vírgula (‘`;`’) ou `\g` ao fim da instrução.

`mysql_query()` não pode ser usadas por consultas que contenham dados binários; você deve utilizar `mysql_real_query()` em seu lugar. (Dados binários podem conter o caracter ‘`\0`’, que `mysql_query()` interpreta como o fim a string de consulta.)

Se você quiser saber se a consulta deve retornar um resultado ou não, você pode utilizar `mysql_field_count()` para verificar isto. Veja Seção 12.1.3.20 [`mysql_field_count()`], Página 797.

Valor Retornado

Zero se a consulta obteve sucesso. Diferente de zero se ocorreu um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.43 mysql_real_connect()

```
MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user,
const char *passwd, const char *db, unsigned int port, const char *unix_socket,
unsigned long client_flag)
```

Descrição

`mysql_real_connect()` tenta estabelecer uma conexão mecanismo MySQL de banco de dados executando em `host`. `mysql_real_connect()` deve completar com suceeso antes que você possa executar qualquer um das outars funçãoes da API, com a excessão de `mysql_get_client_info()`.

Os parâmetros são especificados da seguinte forma:

- O primeiro parâmetro deve ser o endereço de uma estrutura `MYSQL` existente. Antes de chamar `mysql_real_connect()` você deve chamar `mysql_init()` para inicializar a estrutura `MYSQL`. Você pode alterar vária opções de conexão com a chamada `mysql_options()`. Veja Seção 12.1.3.40 [mysql_options], Página 809.
- O valor de `host` pode ser tanto um nome de máquivo quanto um endereço IP. Se `host` é `NULL` ou a string `"localhost"`, a conexão é feita na máquina local. Se o SO suporta sockets (Unix) ou named pipes (Windows), eles são utilizados em vez de TCP/IP para a conexão ao servidor.
- O parâmetro `user` contém a indetificação do usuário MySQL. Se `user` é `NULL` ou a string vazia `""`, considera-se o usuário padrão. Sob Unix, ele é o login atual. Sob ODBC no Windows, o usuário atual deve ser especificado explicitamente. Veja Seção 12.2.2 [Administrador ODBC], Página 871.
- O parâmetro `passwd` contém a senha para `user`. Se `passwd` é `NULL`, somente entradas na tabela `user` para usuários que tenham campo de senha em branco (vazia) serão verificados ipor um padrão coincidenete. Isto permite que o admistrador do banco de dados configure o sistema de privilégios do MySQL de tal maneira que usuários os usuários conseguirão privileios diferentes, dependendo se ele espcificou ou não uma senha.

  Nota: Não tente criptografar a senha antes de chamar `mysql_real_connect()`; senhas criptografadas são tratadas automaticamente pela API cliente.
- `db` é o nome de banco de dados. Se `db` não é `NULL`, a conexão definirá o banco de dados padrão com este valor.
- Se `port` não é 0, o valor será usado como o número da porta para as conexões TCP/IP. Note que o parâmetro `host` determina o tipo da conexão.
- Se `unix_socket` não é `NULL`, a string especifica o socket ou named pipe que deve ser usado. Note que o parâmetro `host` determina o tipo de conexão.
- O valor de `client_flag` é normalmente 0, mas pode ser definido como uma combinação dos parâmetro seguintes em circunstâncias especiais:

Nome do parâmetro	Descrição do parâmetro
`CLIENT_COMPRESS`	Usa protocolo compactado.

`CLIENT_FOUND_ROWS`	Retorna o número de linhas encontradas (correspondentes a um padrão), não o número de linha efetivo.
`CLIENT_IGNORE_SPACE`	Permite espaço depois do nome de funções. torna todos os nomes de funções palavras reservadas.
`CLIENT_INTERACTIVE`	Permite `interactive_timeout` segundos (no lugar de `wait_timeout` segundos) de inatividade antes de fechar a conexão.
`CLIENT_LOCAL_FILES`	Habilita `LOAD DATA LOCAL`.
`CLIENT_MULTI_STATEMENTS`	Diz ao servidor que o cliente pode enviar consultas multi linhas (separado com '`;`'). Se este parâmetro não está definido, consultas de multil linhas está disabilitado. (Novo na versão 4.1).
`CLIENT_MULTI_RESULTS`	Diz ao servidor que o cliente pode tratar multíplos conjuntos de resultados de um multi consulta ou stored procedures. Isto é definido automaticamente se `CLIENT_MULTI_STATEMENTS` está lidado. Novo na versão 4.1.
`CLIENT_NO_SCHEMA`	Não permite a sintaxe `db_name.nome_tabela.nome_coluna`. Isto é para o ODBC. Ele faz com que o analizador gere um erro se você utilizar aquela sintaxe. É útil para achar erros em alguns programas ODBC.
`CLIENT_ODBC`	O cliente é um cliente ODBC. Torna o `mysqld` mais amigável ao ODBC.
`CLIENT_SSL`	Usa SSL (protocolo criptografado). Esta opção não deve ser configuração pelo aplicativo; ele é definida internamente na biblioteca cliente.

Valor Retornado

Um handle de conexão `MYSQL*` se a conexão foi obtida com sucesso, `NULL` se a conexão falhou. Para um conexão estabelecida o valor de retorn é o mesmo que o valor do primeiro parâmetro.

Erros

`CR_CONN_HOST_ERROR`
: Falhou ao conectar ao servidor MySQL.

`CR_CONNECTION_ERROR`
: Falhou ao conectar ao servidor MySQL local.

`CR_IPSOCK_ERROR`
: Falhou au criar um socket IP.

`CR_OUT_OF_MEMORY`
: Sem memória.

`CR_SOCKET_CREATE_ERROR`
: Falhou ao criar um socket Unix.

`CR_UNKNOWN_HOST`
: Falhou ao procurar o endereço IP para o nome de maquina.

`CR_VERSION_ERROR`
: Um erro de protocolo resultou da tentativa de conexao a um servidor com uma biblioteca cliente que utiliza uma versão de protocolo diferente. Isto pode acontecer se você utiliza uma biblioteca cliente muito antiga para se conectar a um novo servidor qua não foi iniciado com a opção `--old-protocol`.

`CR_NAMEDPIPEOPEN_ERROR`
: Falhou ao criar um named pipe no Windows.

`CR_NAMEDPIPEWAIT_ERROR`
: Falhou ao esperar por um named pipe no Windows.

`CR_NAMEDPIPESETSTATE_ERROR`
: Falhou ao conseguir mainpulador do pipe no Windows.

`CR_SERVER_LOST`
: Se `connect_timeout > 0` e leva mais que `connect_timeout` segundos para conectar ao servidor ou se o servidro foi finalizado ao executar o `init-command`.

Exemplo

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"seu_programa");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
    fprintf(stderr, "Failed to connect to database: Error: %s\n",
          mysql_error(&mysql));
}
```

Usando `mysql_options()` a biblioteca MySQL irá ler as seções `[client]` e `[seu_programa]` no arquivo '`my.cnf`' o qual irá assegurar que seu programa irá funcionar, mesmo se alguem tiver configurado o MySQL de um modo fora do padrão.

Note que sob a conexão, `mysql_real_connect()` define o parâmetro `reconnect` (parte da estrutura `MYSQL`) para um valor de `1`. Este parâmetro indica, no evento em que uma consulta não pode ser realizada devido a perda de conexão, para tentar se reconectar ao servidor antes de esgotar as tentativas.

12.1.3.44 `mysql_real_escape_string()`

```
unsigned long mysql_real_escape_string(MYSQL *mysql, char *to, const char
*from, unsigned long length)
```

Descrição

A função é usada para criar um string SQL válida que você pode usar em uma instrução SQL. Veja Seção 6.1.1.1 [Sintaxe de string], Página 469.

A string em `from` é codificada para uma string SQL com escape, levando em conta o conjunto de caracteres atual da conexãon. O resultado é colocada em `to` e uma byte nulo de terminção é adcionado. Caracteres codificados são `NUL` (ASCII 0), '`\n`', '`\r`', '`\`', '`'`', '`"`' e Control-Z (veja Seção 6.1.1 [Literais], Página 469). (O MySQL precisa que apenas a barra invertida e as aspas utilizadas para citar a consulta sejam escapadas. Esta função coloca os outros caracteres entre aspas para torná-lo mais fácil de ser lido em arquivos log.)

A string apontada por `from` deve ter o tamanho de `length` bytes. Você deve alocar o buffer `to` para o tamanho de pelo menos `length*2+1` bytes. (No pior caso, cada caracter pode precisar de ser codificado como se utilizasse dois bytes, e você preciria de espaço para o byte null de terminação.) Quando `mysql_real_escape_string()` retornar, o conteúdo de `to` será uma string terminada em null. O valor é o tamanho da string codificada. não incluindo o caracter nulo usado para terminar a string.

Exemplo

```
char query[1000],*end;

end = strmov(query,"INSERT INTO test_table values(");
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"What's this",11);
*end++ = '\'';
*end++ = ',';
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"binary data: \0\r\n",16);
*end++ = '\'';
*end++ = ')';

if (mysql_real_query(&mysql,query,(unsigned int) (end - query)))
{
   fprintf(stderr, "Failed to insert row, Error: %s\n",
           mysql_error(&mysql));
}
```

A função `strmov()` usada no exemplo está incluída na biblioteca `mysqlclient` e funciona como `strcpy()` mas retorna um ponteiro para null de terminação do primeiro parâmetro.

Valor Retornado

O tamanho do valor colocado em `to`, não incluindo o caracter null de terminação.

Erros

Nenhum.

12.1.3.45 `mysql_real_query()`

```
int mysql_real_query(MYSQL *mysql, const char *query, unsigned long length)
```

Descrição

Executa a consulta SQL apontada por `query`, que deve ser uma string de `length` bytes. A consulta deve consistir de uma instrução SQL simples. Você não deve adicionar um ponto e virgula (‘`;`’) ou `\g` no fim da instrução.

Você **deve** utilizar `mysql_real_query()` em lugar de `mysql_query()` para consultas que contenham dados binários, pois eles podem conter o caracter ‘`\0`’. Além disso, `mysql_real_query()` é mais rápido que `mysql_query()` pois ele não faz chamadas `strlen()` na string de consulta.

Se você quiser saber se a consulta retornou um resultado ou não, você pode usar `mysql_field_count()`. Veja Seção 12.1.3.20 [mysql_field_count], Página 797.

Valor Retornado

Zero se a consulta obteve sucesso. Deiferente de zero se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.46 `mysql_reload()`

```
int mysql_reload(MYSQL *mysql)
```

Descrição

Diz ao servidor MySQL para recarregar a tabela de ables. The connected user must have the `RELOAD` privilege.

This function is deprecated. It is preferable to use `mysql_query()` to issue a SQL `FLUSH PRIVILEGES` statement instead.

Valor Retornado

Zero for success. Non-zero if an error occurred.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.47 `mysql_row_seek()`

```
MYSQL_ROW_OFFSET mysql_row_seek(MYSQL_RES *result, MYSQL_ROW_OFFSET offset)
```

Descrição

Atribui ao cursor de linha um registro arbitrário em resultado de uma consulta. O valor do `offset` é um offset do registro que deve ser um valor retornado de `mysql_row_tell()` ou `mysql_row_seek()`. Este valor não simplesmente um número de linha; se você quiser buscar um registro em um resultado usando o número de linha utilize `mysql_data_seek()`.

Esta função exige que a estrutura do resultado contenha todo o resultado da consulta, assim `mysql_row_seek()` pode ser um usado em conjunto apenas com `mysql_store_result()`, e não com `mysql_use_result()`.

Valor Retornado

O valor anterior do cursor de linha. Este valor pode ser passado a uma chamada subsequente `mysql_row_seek()`.

Erros

Nenhum.

12.1.3.48 `mysql_row_tell()`

```
MYSQL_ROW_OFFSET mysql_row_tell(MYSQL_RES *result)
```

Descrição

Retorna a posição atual do cursor de linha para a última `mysql_fetch_row()`. Este valor pode ser utilizado como argumento para `mysql_row_seek()`.

Você deve utilizar `mysql_row_tell()` somente depois de `mysql_store_result()`, e não depois de `mysql_use_result()`.

Valor Retornado

O offset atual do cursos de linha.

Erros

Nenhum.

12.1.3.49 `mysql_select_db()`

```
int mysql_select_db(MYSQL *mysql, const char *db)
```

Descrição

Faz com que o banco de dados espexcificado por `db` se torne o padrão (atual) na conexão especificada por `mysql`. Nas consultas seguintes este banco de dados é o padrão para tabelas que não incluem uma especificação explicita para o banco de dados.

`mysql_select_db()` falha a menos que o usuário conectado possa ser autenticado com permissão para utilizar o banco de dados.

Valor Retornado

Zero em caso de sucesso. Deiferente de zero se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.50 `mysql_set_server_option()`

```
int mysql_set_server_option(MYSQL *mysql, enum enum_mysql_set_option option)
```

Descrição

Habilita ou desabilita uma opção para a conexão. `option` por ter um dos seguintes valores:

MYSQL_OPTION_MULTI_STATEMENTS_ON — Habilita suporte a multi instruções.
MYSQL_OPTION_MULTI_STATEMENTS_OFF — Desabilita suporte a multi instruções.

Valores Retornados

Zero em caso de sucesso. Deiferente de zero se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`ER_UNKNOWN_COM_ERROR`
: O servidor não suportou `mysql_set_server_option()` (que o caso no qual o servidor é mais antigo que 4.1.1) ou o servidor não suportou a opção que se tentou definir.

12.1.3.51 `mysql_shutdown()`

```
int mysql_shutdown(MYSQL *mysql)
```

Descrição

Diz ao servidor de banco de dados para finalizar. O usuário conectado deve ter privilégio `SHUTDOWN`.

Valor Retornado

Zero em caso de sucesso. Deiferente de zero se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.52 `mysql_sqlstate()`

```
const char *mysql_sqlstate(MYSQL *mysql)
```

Descrição

Retorna uma string terminada em null contendo o código de erro SQLSTATE para o último erro. O código de erro consiste de cinco caracteres. `00000` significa "sem erros". Os valores são especificados pelo ANSI SQL e ODBC. Para uma lista de valores possíveis, veja Seção 13.1 [Error-returns], Página 889.

Note que nem todos os erros já estão mapeados para SQLSTATE. O valor `'HY000'` (erro geral) é usado para erros não mapeados.

Esta função foi adicionada ao MySQL 4.1.1.

Valores Retornados

Uma string terminada em null contendo o código de erro SQLSTATE.

Veja Também

Veja Seção 12.1.3.12 [mysql_errno], Página 791. Veja Seção 12.1.3.13 [mysql_error], Página 792. Veja Seção 12.1.7.18 [mysql_stmt_sqlstate], Página 855.

12.1.3.53 `mysql_ssl_set()`

```
int mysql_ssl_set(MYSQL *mysql, const char *key, const char *cert, const char
*ca, const char *capath, const char *cipher)
```

Descrição

`mysql_ssl_set()` é usado para estabelecer conexão segura usando SSL. Ela deve ser chamada antes de `mysql_real_connect()`.

`mysql_ssl_set()` não faz nada a mesno que o suporte OpenSSL esteja habilitado na biblioteca cliente.

`mysql` e o handler da conexão retornado de `mysql_init()`. Os outros parâmetros são especificados como a seguir:

- `key` é o caminho para o arquivo de chave.
- `cert` é o caminho para o arquivo do certificado.
- `ca` é o caminho para o arquivo de autoridade do certificado.
- `capath` é o caminho para um diretório que contém certificados SSL CA confiáveis no formato pem.
- `cipher` é a lista de cifras permitidas para uso para criptografia SSL.

Qualquer parâmetro SSL não utilizado pode ser dado com `NULL`.

Valores Retornados

Esta função sempre retorna `0`. Se a configuração SSL está incorreta, `mysql_real_connect()` retornará um erro quando você tentar se conectar.

12.1.3.54 mysql_stat()

```
char *mysql_stat(MYSQL *mysql)
```

Descrição

Retorna uma string contendo informações sinmilares a aquelas fornecidas pelo comando `mysqladmin status`. Isto inclui o tempo de conexão em segundos e o número de threads em execução, recargas e tabelas abertas.

Valor Retornado

Uma string descrevendo o status do servidor. `NULL` se um erro ocorrer.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.55 mysql_store_result()

```
MYSQL_RES *mysql_store_result(MYSQL *mysql)
```

Descrição

Você deve chamar `mysql_store_result()` ou `mysql_use_result()` para cada consulta que retorne dados com sucesso (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`).

Você não precisa chamar `mysql_store_result()` ou `mysql_use_result()` para outras consultas, mas ele não causará nenhum dano ou nenhuma queda notel de desempenho se você chamar `mysql_store_result()` em todos os casos. Você pode detectar se a consulta não obteve resultado verificando se `mysql_store_result()` retornou 0.

Se você quiser saber se a consulta devia retornar algum resultado, você pode utilizar `mysql_field_count()` para fazer a verificação. Veja Seção 12.1.3.20 [mysql_field_count], Página 797.

`mysql_store_result()` lê todo o resultado de uma consulta para um cliente, aloca uma estrutura `MYSQL_RES` e coloca o resultado nesta estrutura.

`mysql_store_result()` retorna um ponteiro para null se a consulta não retornar um resultado (se a consulta foi, por exemplo, uma instrução `INSERT`).

`mysql_store_result()` também retorna um ponterio para null se a leitura do resultado falhar. Você pode verficar se você obteve um erro verificando se `mysql_error()` não retornou um ponterio para null, se `mysql_errno()` retorna <> 0, ou se `mysql_field_count()` retorna <> 0.

Um resultado vazio é retornado se não houver registros a retornar. (Um resultado vazio é diferente de um ponteiro para null em um valor de retorno).

Uma vez que você tenha chamado `mysql_store_result()` e tenha retornado um resultado que não é uma apontador para null, você pode chamar `mysql_num_rows()` para descobrir quantas linhas existem no resultado.

Você pode chamar `mysql_fetch_row()` para buscar registros no resultado ou `mysql_row_seek()` e `mysql_row_tell()` para obter ou definir a poição atual do registro dentro do resultado.

Você deve chamar `mysql_free_result()` quando tiver terminado com o resultado.

Veja Seção 12.1.12.1 [`NULL mysql_store_result()`], Página 861.

Valor Retornado

Uma estrutura de resultado `MYSQL_RES` com o resultado. `NULL` se um erro ocorreu.

Erros

`mysql_store_result()` zera `mysql_error` e `mysql_errno` se ela obter sucesso.

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_OUT_OF_MEMORY`
: Sem memoria.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.56 `mysql_thread_id()`

```
unsigned long mysql_thread_id(MYSQL *mysql)
```

Descrição

Retorna a ID da thread da conexão atual. Este valor pode ser usado como um argumento para `mysql_kill()` para finalizar a thread.

Se a conexão for perdida e você reconectar com `mysql_ping()`, a ID da thread irá alterar. Isto significa que você deve obter a ID da thread e guardá-la para uso posterior. Você deve obtê-la quando precisar dela.

Valor Retornado

A ID da thread da conexão atual.

Erros

Nenhum.

12.1.3.57 `mysql_use_result()`

```
MYSQL_RES *mysql_use_result(MYSQL *mysql)
```

Descrição

Você deve chamar `mysql_store_result()` ou `mysql_use_result()` para cada consulta que retornar data com sucesso (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`).

`mysql_use_result()` inicicia a recuperação de um resultado mas não lê realmente o resultado no cliente como `mysql_store_result()` faz. Cada regiostro deve ser recuperado individualmente fazendo chamadas a `mysql_fetch_row()`. Ele lê o resultado de uma consulta diretamente do servidor sem armazenar em uma tabela temporária ou em um buffer local, o o que é mais rápido e utiliza menos memória que `mysql_store_result()`. O cliente sío irá alocar memória para o registro atual para o buffer de comunicação que pode crescer para `max_allowed_packet` bytes.

Por outro lado , você não deve utilizar `mysql_use_result()` se você estiver fazendo vários processamentos para cada registros no lado do cliente, ou se a saída é enviada para a tela, na qual o usuário de digitar um `^S` (parada de tela). Isto irá prender o servidor e impedir outras threads de atualizar qualquer tabela na qual o dados esteja sendo buascado.

Ao usar `mysql_use_result()`, você deve executar `mysql_fetch_row()` até um valor `NULL` ser retornado, senão, os registros não buscados retornarão como part do resultado de sua próxima consulta. A API C fornecerá o erro `Commands out of sync; you can't run this command now` se você esquecer de fazê-lo.

Você não pode utilizar `mysql_data_seek()`, `mysql_row_seek()`, `mysql_row_tell()`, `mysql_num_rows()`, ou `mysql_affected_rows()` com m resultado retornado de `mysql_use_result()`, nem pode executar outras consultas até que `mysql_use_result()` tenha finalizado. (No entanto, depois de buscar todos os regitros, `mysql_num_rows()` retornará corretamente o número de regiostros buscados).

Você deve chamar `mysql_free_result()` após terminar de utilizar o resultado.

Valor Retornado

Uma estrutura de resultado `MYSQL_RES`. `NULL` se ocorrer um erro.

Erros

`mysql_use_result()` zera `mysql_error` e `mysql_errno` se ela obter sucesso.

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_OUT_OF_MEMORY`
: Sem memória.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.3.58 `mysql_warning_count()`

```
unsigned int mysql_warning_count(MYSQL *mysql)
```

Descrição

Retorna o número de avisos gerados durante a execução da instrução SQL anterior. Disponível a partir do MySQL 4.1.

Valores de Retorno

A contagem de avisos.

Errors

Nenhum.

12.1.3.59 `mysql_commit()`

```
my_bool mysql_commit(MYSQL *mysql)
```

Descrição

Faz um commits na transação atual. Disponível no MySQL 4.1

Valor Retornado

Zero em caso de sucesso. Deiferente de zero se ocorrer um erro.

Erros

Nenhum.

12.1.3.60 mysql_rollback()

```
my_bool mysql_rollback(MYSQL *mysql)
```

Descrição

Faz um rollback na transação atual. Disponível no MySQL 4.1

Valor Retornado

Zero em caso de sucesso. Deiferente de zero se ocorrer um erro.

Erros

Nenhum.

12.1.3.61 mysql_autocommit()

```
my_bool mysql_autocommit(MYSQL *mysql, my_bool mode)
```

Descrição

Define o modo autocommit como ligado se `mode` é 1, desligado se `mode` é 0.

Valor Retornado

Zero em caso de sucesso. Deiferente de zero se ocorrer um erro.

Erros

Nenhum.

12.1.3.62 mysql_more_results()

```
my_bool mysql_more_results(MYSQL *mysql)
```

Descrição

Retorna verdade se mais resultados da consulta atualmente em execução existem, e a aplicação deve chamar `mysql_next_result()` para buscar os resultados. Disponível no MySQL 4.1

Valor Retornado

`TRUE` (1) se existem mais resultados. `FALSE` (0) se não existem mais resultados.

Note que na maioria dos casos chama-se `mysql_next_result()` para se mais de um resultado existe e inicia o próximo resultado se ele existir.

Veja Seção 12.1.8 [C API multiple queries], Página 855. Veja Seção 12.1.3.63 [mysql_next_result], Página 827.

Erros

Nenhum.

12.1.3.63 `mysql_next_result()`

```
int mysql_next_result(MYSQL *mysql)
```

Descrição

Se existem mais resultados da consulta, `mysql_next_result()` lê o próximo resultado da consulta e retorna o status a aplicação. Disponível no MySQL 4.1

Note que você deve chamar `mysql_free_result()` para a consulta anterior se ela retornar um resultado.

Depois de chamar `mysql_next_result()` o estado da conexão é como se tivesse chamado `mysql_real_query()` para a prima consulta. Isto significa que você agora pode chamar `mysql_store_result()`, `mysql_warning_count()`, `mysql_affected_rows()` ... na conexão.

Se `mysql_next_result()` retorna um erro, nenhuma outra instrução será executada e não haverá mais resultado para buscar.

Veja Seção 12.1.8 [C API multiple queries], Página 855.

Valor Retornado

0 em caso de sucesso e haver mais resultados. -1 se não houver mais resultados. > 0 se ocorrer um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada. Por exemplo se você não chamar `mysql_use_result()` para um resulatdo anterior.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.4 Instruções Preparadas da API C

A partir da versão 4.1 do MySQL, o protocolo cliente/servidor fornece o uso de instruções preparadas. E capacidade utilizam estruturas de dados de tratamento de instruções `MYSQL_STMT`.

Execução preparada é um modo eficiente de executar uma instrução mais de uma vez. A instrução é primeiramente analizada para prepará-la para a execução. Então é executada uma ou mais vezes posteriormente, utilizando o manipulador de instruções retornado pela função preparada.

Execução preparada é mais rápida que a execução direta para instruções executadas mais que uma vez, pois a consulta é analizada apenas uma vez. No caso de execução direta , a consulta é analisada todas as vezes que ela é executada. Execução preparada também pode fornecer uma redução de tráfico de rede porque para cada execução das instruções preparadas, é necessário enviar dados apenas os parâmetros.

Outra vantagem de instruções preparadas é que ela utiliza um protocolo binário, que faz a transferência dos dados entre clinete e servidor de forma mais eficiente. Instruções preparadas também podem suportar ligação de entrada e saída com a execução de cnsultas múltiplas.

12.1.5 Tipos de Dados de Instruções Preparadas da API C

Note: A API para instruções preparadas ainda é assunto de revisão. Esta informação é fornecida para os adeptos, mas esteja ciente que a API pode alterar.

Instrução preparadas utilizam principalmente as estruturas de dados `MYSQL_STMT` e `MYSQL_BIND` seguintes. Uma terceira estrutura, `MYSQL_TIME`, é usada para tranferir dados temporais.

`MYSQL_STMT`
: Esta estrutura representa uma instrução preparada. Uma instrução é preparada chamando `mysql_prepare()`, que retorna uma handler da instrução, que é um ponteiro para um `MYSQL_STMT`. O handler é usado para todas as funções subsequentes relacionadas às instruções.

  A estrutura `MYSQL_STMT` não possui membros para uso em aplicação.

  Múltiplos handles de instruções podem estar associados com uma única conexão. O limite no número de handlers depende dos recursos de sistemas disponíveis.

`MYSQL_BIND`
: Esta estrutura é usada tanto para a entrada da consulta (valores de dados enviados ao servidor) quanto para saída (valores de resultado retornados do servidor). Para entrada, ela é usada com `mysql_bind_param()` para ligar os valores os dados dos parâmetros para armazenar em buffers para uso pelo `mysql_execute()`. Para saída, ela é usada com `mysql_bind_result()` para ligar o buffer de resultado para uso na busca de registros com `mysql_fetch()`.

A estrutura `MYSQL_BIND` contém os seguintes membros para uso em aplicativos. Cada um deles utiliza tanto a entrada quanto a saída, embora algumas vezes sejam para diferentes propósitos dependendo da direção da transfer6encia de dados:

`enum enum_field_types buffer_type`
: O tipo do buffer. Os valores de `buffer_type` estão listados posteriormente nesta seção. Para entrada, `buffer_type` indica que tipo de valor você está ligando a um parâmetro de uma consulta. Para a saída, ele indica que tipo de valor você espera receber em um buffer de resultado.

`void *buffer`
: Para entrada, este é um ponteiro para o buffer no qual os dados de parâmetros de uma consulta, estão armazenados. Para saída, ele é um ponteiro para o buffer no qual se deve retornar o valor de uma coluna do resultado. Para tipos numéricos, o `buffer` deve apontar para uma variável do tipo C apropriado. (Se você estiver associando a variável com uma coluna que tem o atributo `UNSIGNED`, a variável deve ser um tipo C `unsigned`.) Para colunas de tipo data e hora, o `buffer` deve apontar para uma estrutura `MYSQL_TIME`. Para colunas do tipo caracter e string binária, o `buffer` aponta para um buffer de caracter.

`unsigned long buffer_length`
: O tamanho atual de `*buffer` em bytes. Ele indica a quantidade máxima de dados que pode ser armazenado no buffer. Para caracteres e dados C binários, o valor `buffer_length` especifica o tamanho do `*buffer` quando utilizado com `mysql_bind_param()`, ou o número máximo de bytes de dados que pode ser buscado em um buffer quando usado com `mysql_bind_result()`.

`unsigned long *length`
: Um ponteiro para uma variável `unsigned long` que indica o número atual de bytes de dados armazenado em `*buffer`. `length` é usado é usado para caracteres e dados C binários. Para a ligação dos dados do parâmetro de entrada, `length` aponta para uma variável `unsigned long` que indica o tamanho do valor do parâmetro armazenado em `*buffer`; isto é usado pelo `mysql_execute()`. Se o tamanho é um ponteiro nulo, o protocolo assume que todos os caracteres e dados binários são terminados com null.

  Para ligação dos valores de saída, `mysql_fetch()` coloca o tamanho dos valores de coluna retornados na variável para onde o `*length` aponta.

  `length` é ignorado por tipos de dados numéricos e trmporais porque o tamanho do valord dos dados é determinado pelo valor `buffer_type`.

`bool *is_null`
: Este membro aponta para uma variável `my_bool` que é verdadeiro se um valor é `NULL`, falso se ele não é `NULL`. Para entrada, defina `*IS_NULL` como verdadeiro para indicar que você está passando um valor `NULL` como um parâmetro. Para saída, este valor é verdadeiro se o valor de um resultado retornado de uma consulta é `NULL`.

`MYSQL_TIME`
: Esta estrutura é utilizada para enviar e receber dados `DATE`, `TIME`, `DATETIME` e `TIMESTAMP` diretamente de e para o servidor. Isto é feito configurando o membro `buffer_type` de uma estrutura `MYSQL_BIND` para um dos tipos temporais e configurando o membro `buffer` para apontar para uma estrutura `MYSQL_TIME`.

  A estrutura `MYSQL_TIME` contém os seguintes membros:

  `unsigned int year`
  : O ano.

  `unsigned int month`
  : O mês do ano.

  `unsigned int day`
  : O dia do mês.

  `unsigned int hour`
  : A hora do dia.

  `unsigned int minute`
  : O minuto da hora.

  `unsigned int second`
  : Os segundos.

  `my_bool neg`
  : Um parâmetrio booleano para indicar se o tempo é negativo.

  `unsigned long second_part`
  : A parte fracionária do segundo. Este membro não é atualmente usado.

  Apenas aquelas partes de uma estrutura `MYSQL_TIME` que se aplica a um dado tipo de valor temporal são usados: Os elementos `year`, `month` e `day` são usados para valores `DATE`, `DATETIME` e `TIMESTAMP`. Os elementos `hour`, `minute` e `second` são usados para valores `TIME`, `DATETIME` e `TIMESTAMP`. Veja Seção 12.1.9 [C API date handling], Página 856.

A seguinte tabela mostra os valores permitidos que podem ser especificados no membro `buffer_type` da estrutura `MYSQL_BIND`. A tabela também mostra aqueles tipos SQL que correspondem mais proximamente a cada valor `buffer_type`, e, para tipos numéricos e temporais, o tipo C correspondente.

`buffer_type` **Valor**	**Tipo SQL**	**Tipo C**
`MYSQL_TYPE_TINY`	`TINYINT`	`char`
`MYSQL_TYPE_SHORT`	`SMALLINT`	`short int`

MYSQL_TYPE_LONG	INT	long int
MYSQL_TYPE_LONGLONG	BIGINT	long long int
MYSQL_TYPE_FLOAT	FLOAT	float
MYSQL_TYPE_DOUBLE	DOUBLE	double
MYSQL_TYPE_TIME	TIME	MYSQL_TIME
MYSQL_TYPE_DATE	DATE	MYSQL_TIME
MYSQL_TYPE_DATETIME	DATETIME	MYSQL_TIME
MYSQL_TYPE_TIMESTAMP	TIMESTAMP	MYSQL_TIME
MYSQL_TYPE_STRING	CHAR	
MYSQL_TYPE_VAR_STRING	VARCHAR	
MYSQL_TYPE_TINY_BLOB	TINYBLOB/TINYTEXT	
MYSQL_TYPE_BLOB	BLOB/TEXT	
MYSQL_TYPE_MEDIUM_BLOB	MEDIUMBLOB/MEDIUMTEXT	
MYSQL_TYPE_LONG_BLOB	LONGBLOB/LONGTEXT	

Uma conversão de tipo implcita pode ser realizada em ambas as direções.

12.1.6 Visão Geral das Funções de Instruções Preparadas da API C

Note: A API para instruções preparadas ainda é assunto de revisão. Esta informação é fornecida para os adeptos, mas esteja ciente que a API pode alterar.

As funções disponíveis nas instruções preparadas estão resumidas aqui e desctias em maiores detalhes em um seção posterior. Veja Seção 12.1.7 [C API Prepared statement functions], Página 833.

Função	Descrição
mysql_prepare()	Prepara uma string SQL para execução.
mysql_param_count()	Retorna o número de parâmetros em uma instrução SQL preparada.
mysql_get_metadata()	Retorna metadados de instruções preparadas em forma de um conjunto de resultados.
mysql_bind_param()	Associa o buffers de dados da aplicação com o parâmetro marcado na instrução SQL preparada.
mysql_execute()	Executa a instrução preparada.
mysql_stmt_affected_rows()	Retorna o número de registros alteradosi, deletados ou inseridos pela última consulta `UPDATE`, `DELETE`, ou `INSERT`.
mysql_bind_result()	Associa o buffers de dados da aplicação com colunas no resultado.
mysql_stmt_store_result()	Retorna o resultado completo para o cliente.
mysql_stmt_data_seek()	Busca um número de registro arbitrário no resultado de uma consulta.

mysql_stmt_row_seek()	Busca por um offset de registro no resultado de uma busca, utilizando o valor reotornado de `mysql_stmt_row_tell()`.
mysql_stmt_row_tell()	Retorna a posição do cursor de registro.
mysql_stmt_num_rows()	Retorna o total de registros do resultado de uma instrução armazenada.
mysql_fetch()	Busca o próximo conjunto de dados do resultado e retorna os dados para todas as colunas limites.
mysql_stmt_close()	Libera a memória usada pela instrução preparada.
mysql_stmt_errno()	Retorna o número de erro para a última instrução executada.
mysql_stmt_error()	Retorna a mensagem de erro para a última instrução executada.
mysql_stmt_sqlstate()	Retorna o código de erro SQLSTATE para a execução da última instrução.
mysql_send_long_data()	Envia dados longos em blocos para o servidor.

Chama `mysql_prepare()` para preparar e iniciar o manipulador de instruções, `mysql_bind_param()` para fornecer os dados do parâmetro e `mysql_execute()` para executar a consulta. Você pode repetir o `mysql_execute()` alterando o valor do parâmetro no buffer respectivo fornecido por `mysql_bind_param()`.

Se a consulta é uma instrução `SELECT` ou qualquer outra consulta que produz um resultado, `mysql_prepare()` também retornará a informação dos meta dados do resultado na forma de um resultado `MYSQL_RES` através de um `mysql_get_metadata()`.

Você pode forncer o buffer de resultado usando `mysql_bind_result()`, assim `mysql_fetch()` retornará automaticamente os dados para este buffer. Esta busca é feita registro a registro.

Você também pode enviar o texto ou dado binário em blocos para o servidor utilizando `mysql_send_long_data()`, especficando a opção `is_long_data=1` ou `length=MYSQL_LONG_DATA` ou `-2` na estrutura `MYSQL_BIND` fornecida com `mysql_bind_param()`.

Quando a execução for completada, o handler da instrução deve ser fechado usando `mysql_stmt_close()` para que todos os recursos associados a ele sejam liberados.

Se você obteve os metadados de um resultado de uma instrução `SELECT` chamando `mysql_get_metadata()`, você também deve liberá-lo usando `mysql_free_result()`.

Execution Steps:

Para prepara e executar uma instrução, uma aplicação:

1. Chama `mysql_prepare()` e passa uma string contendo uma instrução SQL. Em uma operação de preparo bem sucedida, o `mysql_prepare` retorna o manipulador de instrução válido para a aplicação.

2. Se a consulta produz um resultado, chama `mysql_get_metadata` para obter o conjunto de resultado de metadados. Este metadado está na forma de um resultado, embora um separado daqueles que contém as linhas retornadas pela consulta. O resultado de metadados indica quantos colunas estão no resultado e contém informações sobre cada coluna.
3. Define o valor de qualquer parâmetro usando `mysql_bind_param`. Todos os parâmetros devem ser definidos. De outra forma a execução da consulta retornará um erro ou produzirá resultados inesperados.
4. Chama `mysql_execute()` para executar a instrução.
5. Se a consulta produz um resultado, liga o buffer de dados usado para retornar o valor do registro chamando `mysql_bind_result()`.
6. Busca os dados no buffer, registro a registro chamando `mysql_fetch()` repetidas vezes até não haver mais registros.
7. Repete os passos de 3 a 6 como necessário, alterando o valor dos parâmetros e re-executando a instrução.

Quando `mysql_prepare()` é chamado, o protocolo cliente/servidor do MySQL realiza as seguintes ações:

- O servidor analiza a consulta e envia o status de OK de volta para o cliente atribuindo uma identificação de instrução. Ele também envia um número total de parâmetros, uma contagem de colunas e sua meta informação se for um resultado orientado a consulta. Toda a sintaxe e semântica da consulta é verificada pelo servidor durante a chamada.
- O cliente utiliza esta identificação da instrução para as operações adicionais, assim o servidor pode identificar a instrução dentre outras existentes. O cliente também aloca um manipulador de instruções com esta identificação e o retorna para a aplicação.

Quando o **mysql_execute()** é chamado, no protocolo cliente/servidor do MySQL realiza as seguintes operações:

- O cliente utiliza o manipulador de instruções e envia o dado do parâmetro para o servidor.
- O servidor identifica a instrução usando a identificação fornecida pelo cliente, substitui o marcador do parâmetro com o dado fornecido mais recente e executa a consulta. Se a consulta produz um resultado, o servidor envia o dado de volta para o cliente. Senão envia o status de OK como número total de registros alterados, deletados ou inseridos.

Quando `mysql_fetch()` é chamado, no protocolo cliente/servidor do MySQL realiza as seguintes ações:

- O cliente lê os dados do pacote registro por registro e o coloca no buffer de dados da aplicação fazendo as conversões necessárias. Se o tipo do buffer de aplicação é o mesmo do tipo do campo retornado do servidor, as conversões são diretas.

Você pode obter o código de erro, mensagens e o valor SQLSTATE da instrução utilizando `mysql_stmt_errno()`, `mysql_stmt_error()` e `mysql_stmt_sqlstate()` respectivamente.

12.1.7 Descrição das Funções de Instrução Preparada da API C

Para preparar e executar consultas use as seguites funçoes.

12.1.7.1 mysql_prepare()

```
MYSQL_STMT * mysql_prepare(MYSQL *mysql, const char *query, unsigned long
length)
```

Descrição

Prepara a consulta SQL apontada pela string com terminação em nulo `query`, e retorna um handle da instrução para ser usado por operações adicionais na instrução. A consulta deve consistir de uma única instrução SQL. Você não deve adicionar ponto e virgula (‘`;`’) ou \g a instrução.

A aplicação pode incluir um ou mais marcadores de parâmetro na instrução SQL, embutindo interrogações (‘`?`’) na string SQL na posição aprpriada.

Os marcadores só são válidos em certos lugares na instrução SQL. Por exemplo, eles não são permitidos em lista `VALUES()` de uma instrução `INSERT` (para especificar valores para uma linha ou em uma comparação com uma coluna em uma cláusula `WHERE` para especificar uma valor de comparação. No entanto, eles não são permitidos como identificadores (tais como nomes de colunas ou tabelas), na lista select que indica as colunas a serem retornadas por uma instrução `SELECT`), ou para especificar ambos operandos de um operador binário como o sinal de igual `=`. A última restrição é necessária porque seria impossível determinar o tipo do parâmetro. Em geral, parâmetros são válidos somente em instrução de Linguagem de Manipulação de Dados (Data Manipulation Languange-DML), e não em instruções de Linguagem de Definição de Dados (Data Defination Language-DDL).

Os marcadores de parâmetro devem limitar variáveis de aplicações utilizando `mysql_bind_param()` antes de executar a instrução.

Valor Retornado

Um ponteiro para uma estrutura `MYSQL_STMT` se o preparo obteve sucesso. `NULL` se ocorreu um erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_OUT_OF_MEMORY`
: Falta de memória

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

Se o preparo não obteve sucesso (isto é, `mysql_prepare()` retorna um ponteiro `NULL`), as mensagens de erros podem ser obtidas chamando `mysql_error()`.

Exemplo

Para o uso de `mysql_prepare()` consulte o exemplo de Seção 12.1.7.5 [`mysql_execute()`], Página 837.

12.1.7.2 `mysql_param_count()`

```
unsigned long mysql_param_count(MYSQL_STMT *stmt)
```

Descrição

Retorna o número de marcadores de parâmetros presentes na consulta preparada.

Valor Retornado

Um unsigned long (inteiro sem sinal) representando o número de parâmetros em uma instrução.

Erros

Nenhum.

Exemplo

Para utilizar `mysql_param_count()` consulte o exemplo de Seção 12.1.7.5 [`mysql_execute()`], Página 837.

12.1.7.3 `mysql_get_metadata()`

```
MYSQL_RES *mysql_get_metadata(MYSQL_STMT *stmt)
```

Descrição

Se uma instrução passada para `mysql_prepare()` rproduziu um resultado, `mysql_get_metadata()` retorna o resultado dos meta dados na forma de um ponteiro para uma estrutura `MYSQL_RES` que também pode ser usada para processar a meta informação como o número total de campos e informação de campos indivíduais. Este ponteriro para o resultado pode ser passado como um argumento para qualquer um dos campos com base na API que processam o resultado dos metadados, como:

- `mysql_num_fields()`
- `mysql_fetch_field()`

- `mysql_fetch_field_direct()`
- `mysql_fetch_fields()`
- `mysql_field_count()`
- `mysql_field_seek()`
- `mysql_field_tell()`
- `mysql_free_result()`

A estrutura do resultado deve estar liberada quando você acabar de usá-lo. Você pode fazê-lo passando para `mysql_free_result()`. É semelhante ao modo que você libera um resulatdo chamado com `mysql_store_result()`.

O resultado retornado por `mysql_get_metadata()` contém apenas metadados. Ele não contém qualquer resultado de registro. As linhas são obtidas usando o handle de instrução com `mysql_fetch()`.

Valor Retornado

Uma estrutura de resultado `MYSQL_RES`. `NULL` se nenhuma meta informação existe para a consulta preparada.

Erros

`CR_OUT_OF_MEMORY`
: Falta de memória

`CR_UNKNOWN_ERROR`
: Ocorreu um erro desconhecido

Exemplo

Para utilizar `mysql_get_metadata()` consulte o exemplo de Seção 12.1.7.13 [`mysql_fetch()`], Página 846

12.1.7.4 `mysql_bind_param()`

```
my_bool mysql_bind_param(MYSQL_STMT *stmt, MYSQL_BIND *bind)
```

Descrição

`mysql_bind_param()` é utilizado para ligar dados para os marcadores de parâmetros na instrução SQL que foi passada para `mysql_prepare()`. Ele utiliza a estrutura `MYSQL_BIND` para fornecer os dados. `bind` é o endereço de um vetor de estruturas `MYSQL_BIND`. A biblioteca cliente espera que o vetor deve contenha um elemento para cada marcador de parâmetro `?` que está presente na consulta.

Suponha que você prepare a seguinte instrução:

```
INSERT INTO mytbl VALUES(?,?,?)
```

Quando você ligar os parâmetros, o vetor da estrutura `MYSQL_BIND` deve conter três elementos e pode estar declarado assim:

```
MYSQL_BIND bind[3];
```

O membro de cada elemento `MYSQL_BIND` que deve estar configurado está descrito em Seção 12.1.5 [C API Prepared statement datatypes], Página 828.

Valor Retornado

Zeros se a ligação foi obtida com sucesso. Diferente de zero se ocorrer um erro.

Erros

`CR_NO_PREPARE_STMT`
: Não existem instruções preparadas

`CR_NO_PARAMETERS_EXISTS`
: Não existem parâmetros para ligar

`CR_INVALID_BUFFER_USE`
: Indica se a ligação forncerá dados longos em bolcos e se o tipo de buffer é binário ou não é uma string.

`CR_UNSUPPORTED_PARAM_TYPE`
: A conversão não é suportada. Possivelmente o valor de `buffer_type` é inválido ou não é um dos tipos suportados listados acima.

`CR_OUT_OF_MEMORY`
: Falta de memória

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

Exemplo

Para utilizar `mysql_bind_param()` consulte o exemplo de Seção 12.1.7.5 [`mysql_execute()`], Página 837.

12.1.7.5 `mysql_execute()`

`int mysql_execute(MYSQL_STMT *stmt)`.

Descrição

`mysql_execute()` executa a consulta preparada associada ao controlador de instruções. O valor atual do marcador de parâmetros é enviado para o servidor durante esta chamada, e o servidor substituir marcadores com os novos dados fornecidos.

Se a instrução é um `UPDATE`, `DELETE` ou `INSERT`, o número total de registros altrados, deletados ou inseridos pode ser encontrado chamando `mysql_stmt_affected_rows()`. Se este é um resultado de uma consulta como `SELECT`, deve se chamar `mysql_fetch()` para buscar dados previamente para fazer qualquer outra função que resulte em um processamento de consulta. Para mais informações sobre como buscar os resultados, consulte Seção 12.1.7.13 [`mysql_fetch()`], Página 846

Valor Retornado

Zero se a execução obteve sicesso. Diferente de zero se ocorreu um erro. O código de erro e a mensagem podem ser obtidas chamando `mysql_stmt_errno()` e `mysql_stmt_error()`.

Erros

`CR_NO_PREPARE_QUERY`
: Nenhuma consulta preprada previamente para execução

`CR_ALL_PARAMS_NOT_BOUND`
: Não forma fornecidos todos os dados de parâmetros.

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_OUT_OF_MEMORY`
: Falta de memória

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

Exemplo

O seguinte exemplo demonstra como criar e preencher uma tabela usando `mysql_prepare()`, `mysql_param_count()`, `mysql_bind_param()`, `mysql_execute()` e `mysql_stmt_affected_rows()`. A variável `mysql` é considerada como um controlador de conexão válido.

```
#define STRING_SIZE 50

#define DROP_SAMPLE_TABLE "DROP TABLE IF EXISTS test_table"
#define CREATE_SAMPLE_TABLE "CREATE TABLE test_table(col1 INT,\
                                                     col2 VARCHAR(40),\
                                                     col3 SMALLINT,\
                                                     col4 TIMESTAMP)"
#define INSERT_SAMPLE "INSERT INTO test_table(col1,col2,col3) VALUES(?,?,?)"
```

```
MYSQL_STMT    *stmt;
MYSQL_BIND    bind[3];
my_ulonglong  affected_rows;
int           param_count;
short         small_data;
int           int_data;
char          str_data[STRING_SIZE];
unsigned long str_length;
my_bool       is_null;

if (mysql_query(mysql, DROP_SAMPLE_TABLE))
{
  fprintf(stderr, " DROP TABLE failed\n");
  fprintf(stderr, " %s\n", mysql_error(mysql));
  exit(0);
}

if (mysql_query(mysql, CREATE_SAMPLE_TABLE))
{
  fprintf(stderr, " CREATE TABLE failed\n");
  fprintf(stderr, " %s\n", mysql_error(mysql));
  exit(0);
}

/* Prepare an INSERT query with 3 parameters */
/* (the TIMESTAMP column is not named; it will */
/* be set to the current date and time) */
stmt = mysql_prepare(mysql, INSERT_SAMPLE, strlen(INSERT_SAMPLE));
if (!stmt)
{
  fprintf(stderr, " mysql_prepare(), INSERT failed\n");
  fprintf(stderr, " %s\n", mysql_error(mysql));
  exit(0);
}
fprintf(stdout, " prepare, INSERT successful\n");

/* Get the parameter count from the statement */
param_count= mysql_param_count(stmt);
fprintf(stdout, " total parameters in INSERT: %d\n", param_count);

if (param_count != 3) /* validate parameter count */
{
  fprintf(stderr, " invalid parameter count returned by MySQL\n");
  exit(0);
}
```

```
/* Bind the data for all 3 parameters */

/* INTEGER PARAM */
/* This is a number type, so there is no need to specify buffer_length */
bind[0].buffer_type= MYSQL_TYPE_LONG;
bind[0].buffer= (char *)&int_data;
bind[0].is_null= 0;
bind[0].length= 0;

/* STRING PARAM */
bind[1].buffer_type= MYSQL_TYPE_VAR_STRING;
bind[1].buffer= (char *)str_data;
bind[1].buffer_length= STRING_SIZE;
bind[1].is_null= 0;
bind[1].length= &str_length;

/* SMALLINT PARAM */
bind[2].buffer_type= MYSQL_TYPE_SHORT;
bind[2].buffer= (char *)&small_data;
bind[2].is_null= &is_null;
bind[2].length= 0;

/* Bind the buffers */
if (mysql_bind_param(stmt, bind))
{
  fprintf(stderr, " mysql_bind_param() failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Specify the data values for the first row */
int_data= 10;             /* integer */
strncpy(str_data, "MySQL", STRING_SIZE); /* string  */
str_length= strlen(str_data);

/* INSERT SMALLINT data as NULL */
is_null= 1;

/* Execute the INSERT statement - 1*/
if (mysql_execute(stmt))
{
  fprintf(stderr, " mysql_execute(), 1 failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Get the total number of affected rows */
```

```
affected_rows= mysql_stmt_affected_rows(stmt);
fprintf(stdout, " total affected rows(insert 1): %ld\n", affected_rows);

if (affected_rows != 1) /* validate affected rows */
{
  fprintf(stderr, " invalid affected rows by MySQL\n");
  exit(0);
}

/* Specify data values for second row, then re-execute the statement */
int_data= 1000;
strncpy(str_data, "The most popular open source database", STRING_SIZE);
str_length= strlen(str_data);
small_data= 1000;         /* smallint */
is_null= 0;               /* reset */

/* Execute the INSERT statement - 2*/
if (mysql_execute(stmt))
{
  fprintf(stderr, " mysql_execute, 2 failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Get the total rows affected */
affected_rows= mysql_stmt_affected_rows(stmt);
fprintf(stdout, " total affected rows(insert 2): %ld\n", affected_rows);

if (affected_rows != 1) /* validate affected rows */
{
  fprintf(stderr, " invalid affected rows by MySQL\n");
  exit(0);
}

/* Close the statement */
if (mysql_stmt_close(stmt))
{
  fprintf(stderr, " failed while closing the statement\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}
```

Nota: Para exemplos completos do uso das funções de instruções preparadas, veja '`tests/client_test.c`'. Este arquivo pode ser obtido em uma distribuição fonte ou do repositório do Bitkeeper.

12.1.7.6 mysql_stmt_affected_rows()

```
my_ulonglong mysql_stmt_affected_rows(MYSQL_STMT *stmt)
```

Descrição

Retorna o número total de registros alterados, deletados ou inseridos pela última instrução executada. Pode ser chamada imediatamente depois de `mysql_execute()` para instruções `UPDATE`, `DELETE` ou `INSERT`. Para instruções `SELECT`, `mysql_stmt_affected_rows()` funciona como `mysql_num_rows()`.

Valor Retornado

Um integer (inteiro) maior que zero indica o número de registros afetados ou retornados. Zero indica que nenhum registro foi atualizado em uma instrução `UPDATE`, nenhum regitro coincidiu com a cláusula `WHERE` na consulta ou que nenhuma consulta foi exeutada ainda. −1 indica que a consulta retornou um erro ou que, para uma consulta `SELECT`, `mysql_stmt_affected_rows()` foi chamado antes de chamra `mysql_fetch()`.

Erros

Nenhum.

Exemplo

Para utilizar `mysql_stmt_affected_rows()` consulte o exemplo de Seção 12.1.7.5 [`mysql_execute()`], Página 837.

12.1.7.7 mysql_bind_result()

```
my_bool mysql_bind_result(MYSQL_STMT *stmt, MYSQL_BIND *bind)
```

Descrição

`mysql_bind_result()` é usado para associar (ligar) colunas no resultados ao buffer de dados e buffer de tamanho. Quando `mysql_fetch()` é chamado para buscar dados, o protocolo cliente/servidor MySQL os coloca os dados para as colunas limite no buffer especificado.

Note que todas as colunas devem ser limitadas por buffers antes da chamada de `mysql_fetch()`. `bind` é o endereço de um vetor de estruturas `MYSQL_BIND`. A biblioteca cliente espera que o vetor contenha um elemento para cada coluna no resultado. Senão `mysql_fetch()` simplesmente ignorado os dados trazidos; os buffers devem se suficientemente grande para guardar os dados, porque o protocolo não retorna dados em blocos.

Uma coluna pode ser limitada a qualquer hora, mesmo depois do resultado ter sido parcialmente recuperado. A nova ligação tem efeito ba próxima vez em que `mysql_fetch()`

é chamado. Suponha que uma aplicação liga a coluna em um resultado e chama `mysql_fetch()`. O protocolo cliente/servidor retorna dados em buffers limitados. Agora suponha que a aplicação ligue a coluna a um diferente cojunto de buffers, então o protocolo não coloca os dados em um novo buffer limitado até que a próxima chamada `mysql_fetch()` ocorra.

Para ligar uma coluna, uma aplicação chama `mysql_bind_result()` e passa o tipo, o endereço e o endereço do buffer do tamanho. Os membros de cada elemento `MYSQL_BIND` que deve ser configurado estão descritos em Seção 12.1.5 [C API Prepared statement datatypes], Página 828.

Valor Retornado

Zero se a ligação obteve sucesso. Diferente de zero se ocorreu um erro.

Erros

`CR_NO_PREPARE_STMT`
: Não existe instruções preparadas

`CR_UNSUPPORTED_PARAM_TYPE`
: A conversão não é suportada. Possivelmente o `buffer_type` é inválido ou não na lista dos tipos de buffers suportados

`CR_OUT_OF_MEMORY`
: Falta de memória

`CR_UNKNOWN_ERROR`
: Ocorreu um erro desconhecido

Exemplo

Para utilizar `mysql_bind_result()` consulta o exemplo de Seção 12.1.7.13 [`mysql_fetch()`], Página 846

12.1.7.8 `mysql_stmt_store_result()`

```
int mysql_stmt_store_result(MYSQL_STMT *stmt)
```

Descrição

Você deve chamar `mysql_stmt_store_result()` para cada consulta que produz um resultado com sucesso (`SELECT`,`SHOW`,`DESCRIBE`, `EXPLAIN`), e só se você quiser armazenar todo o resultado no buffer no cliente, assim que a chamada `mysql_fetch()` subsequente retornar os dados em buffers.

Você é necessário chamar `mysql_stmt_store_result()` para outras consultas, mas se você o fizer, não causará nenhum dano ou queda de performance em todo caso. Você pode detectar se a consulta produziu um resultado verificado se `mysql_get_metadata()` retorna `NULL`. Para mais informações consulte Seção 12.1.7.3 [`mysql_get_metadata()`], Página 835.

Valor Retornado

Zero se o resultado foi armazenado em buffer com sucesso ou Diferente de zero em caso de erro.

Erros

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_OUT_OF_MEMORY`
: Falta de memoria.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_SERVER_LOST`
: A conexão ao servidor MySQL foi perdida durante a consulta.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

12.1.7.9 `mysql_stmt_data_seek()`

```
void mysql_stmt_data_seek(MYSQL_STMT *stmt, my_ulonglong offset)
```

Descrição

Busca um registro arbitrário no resultado de uma instrução. O valor do `offset` é um número de registro e deve estar na faixa de 0 a `mysql_stmt_num_rows(stmt)-1`.

Esta função exige que a estrutura do resultado da instrução contenha todo o resultado da última consulta executada, assim `mysql_stmt_data_seek()` pode ser usada em conjunto apenas com `mysql_stmt_store_result()`.

Valor Retornado

Nenhum.

Erros

Nenhum.

12.1.7.10 `mysql_stmt_row_seek()`

```
MYSQL_ROW_OFFSET mysql_stmt_row_seek(MYSQL_STMT *stmt, MYSQL_ROW_OFFSET
offset)
```

Descrição

Define o cursor de linha com um registro arbitrário em um resultado de instrução. O valor do `offset` é um offset de registro e deve ser um valor retornado de `mysql_stmt_row_tell()` ou `mysql_stmt_row_seek()`. Este valor não é um número de linha; se você quiser buscar um registro em um resultado usando um número de linha, utilize `mysql_stmt_data_seek()`.

Esta função exige que a estrutura do resultado contenha todo o resultado da consulta, assim `mysql_stmt_row_seek()` pode ser usado em conjunto apenas com `mysql_stmt_store_result()`.

Valor Retornado

O valor anterior do cursor de linha. Este valor pode ser passado a uma chamada subsequente de `mysql_stmt_row_seek()`.

Erros

Nenhum.

12.1.7.11 `mysql_stmt_row_tell()`

```
MYSQL_ROW_OFFSET mysql_stmt_row_tell(MYSQL_STMT *stmt)
```

Descrição

Retorna a posição corrente do cursor de linha para o último `mysql_fetch()`. Este valor pode ser usado como um argumento para `mysql_stmt_row_seek()`.

Você deve usar `mysql_stmt_row_tell()` somente depois de `mysql_stmt_store_result()`.

Valor Retornado

O offset atual do cursor de linha.

Erros

Nenhum.

12.1.7.12 `mysql_stmt_num_rows()`

```
my_ulonglong mysql_stmt_num_rows(MYSQL_STMT *stmt)
```

Descrição

Rertorna o número de registros no resultado.

O uso de `mysql_stmt_num_rows()` depende de se você utilizou ou não `mysql_stmt_store_result()` para armazenar todo o resultado no manipulador de instruções.

Se você utilizou `mysql_stmt_store_result()`, `mysql_stmt_num_rows()` pode ser chamado imediatamente.

Valor Retornado

O número de linhas no resultado.

Erros

Nenhum.

12.1.7.13 `mysql_fetch()`

```
int mysql_fetch(MYSQL_STMT *stmt)
```

Descrição

`mysql_fetch()` retorna o próximo registro no resultado. Ele pode ser chamado apenas enquanto existir o conjunto de resultados. Per exemplo, depois de uma chamada de `mysql_execute()` que cria o resultado ou depois de `mysql_stmt_store_result()`, que é chamado depois de `mysql_execute()` para armazenar todo o resultado.

`mysql_fetch` retorna os dados de uma linha usando o buffers limitado por `mysql_bind_result()`. Ele retorna os dados neste buffer para todas as colunas no registro atual e os tamanhos são retornados para o apontador `length`.

Note que, todas as colunas devem ser limitadas pela aplicação antes de chamar `mysql_fetch()`.

Se um valor do dado buscado é um valor `NULL`, o valor `*is_null` da estrutura `MYSQL_BIND` correspondente contém VERDADEIRO (1). Senão, o dado e seu tamanho é retornado nos elementos `*buffer` e `*length` baseados no tipo de buffer especificado pela aplicação. Cada tipo numérico e temporal tem um tamanho fixo como mostrado na tabela a seguir. O tamano dos tipos strings dependem do tasmanho do valor dos dados atual, como indicado por `data_length`.

Type	Length
MYSQL_TYPE_TINY	1
MYSQL_TYPE_SHORT	2
MYSQL_TYPE_LONG	4
MYSQL_TYPE_LONGLONG	8
MYSQL_TYPE_FLOAT	4
MYSQL_TYPE_DOUBLE	8

MYSQL_TYPE_TIME	sizeof(MYSQL_TIME)
MYSQL_TYPE_DATE	sizeof(MYSQL_TIME)
MYSQL_TYPE_DATETIME	sizeof(MYSQL_TIME)
MYSQL_TYPE_TIMESTAMP	sizeof(MYSQL_TIME)
MYSQL_TYPE_STRING	tam_dado
MYSQL_TYPE_VAR_STRING	tam_dado
MYSQL_TYPE_TINY_BLOB	tam_dado
MYSQL_TYPE_BLOB	tam_dado
MYSQL_TYPE_MEDIUM_BLOB	tam_dado
MYSQL_TYPE_LONG_BLOB	tam_dado

Valor Retornado

Valor retornado	Descrição
0	Sucesso, o dado foi buscado para o buffers de dados da aplicação.
1	Ocorreu um erro. O código e a mensagem de erro podem ser obtidos chamando `mysql_stmt_errno()` e `mysql_stmt_error()`.
MYSQL_NO_DATA	Não existem mais registros/dados

Erros

CR_COMMANDS_OUT_OF_SYNC
: Os comando foram executados em uma ordem inpropriada.

CR_OUT_OF_MEMORY
: Falta de memoria.

CR_SERVER_GONE_ERROR
: O servidor MySQL foi finalizado.

CR_SERVER_LOST
: A conexão ao servidor MySQL foi perdida durante a consulta.

CR_UNKNOWN_ERROR
: Um erro desconhecido ocorreu.

CR_UNSUPPORTED_PARAM_TYPE
: O tipo de buffer é MYSQL_TYPE_DATE, MYSQL_TYPE_TIME, MYSQL_TYPE_DATETIME, ou MYSQL_TYPE_TIMESTAMP, mas o tipo de dado não é DATE, TIME, DATETIME ou TIMESTAMP.

  Todos os outros erros de conversão não suportada são retornados de `mysql_bind_result()`.

Exemplo

O seguinte exemplo demonstra como buscar dados de uma tabela usando `mysql_get_metadata()`, `mysql_bind_result()` e `mysql_fetch()`. (Este exemplo espera recuperar

as duas linahs inseridas pelo exemplo mostrado em Seção 12.1.7.5 [mysql_execute()], Página 837.) A variável mysql çonsiderada como um handle de conexão válido handle.

```
#define STRING_SIZE 50

#define SELECT_SAMPLE "SELECT col1, col2, col3, col4 FROM test_table"

MYSQL_STMT    *stmt;
MYSQL_BIND    bind[4];
MYSQL_RES     *prepare_meta_result;
MYSQL_TIME    ts;
unsigned long length[4];
int           param_count, column_count, row_count;
short         small_data;
int           int_data;
char          str_data[STRING_SIZE];
my_bool       is_null[4];

/*
  Sample which is incorporated directly in the manual under Prepared
  statements section (Example from mysql_fetch()
*/

/* Prepare a SELECT query to fetch data from test_table */
stmt = mysql_prepare(mysql, SELECT_SAMPLE, strlen(SELECT_SAMPLE));
if (!stmt)
{
  fprintf(stderr, " mysql_prepare(), SELECT failed\n");
  fprintf(stderr, " %s\n", mysql_error(mysql));
  exit(0);
}
fprintf(stdout, " prepare, SELECT successful\n");

/* Get the parameter count from the statement */
param_count= mysql_param_count(stmt);
fprintf(stdout, " total parameters in SELECT: %d\n", param_count);

if (param_count != 0) /* validate parameter count */
{
  fprintf(stderr, " invalid parameter count returned by MySQL\n");
  exit(0);
}

/* Fetch result set meta information */
prepare_meta_result = mysql_get_metadata(stmt);
if (!prepare_meta_result)
{
```

```
  fprintf(stderr, " mysql_get_metadata(), returned no meta information\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Get total columns in the query */
column_count= mysql_num_fields(prepare_meta_result);
fprintf(stdout, " total columns in SELECT statement: %d\n", column_count);

if (column_count != 4) /* validate column count */
{
  fprintf(stderr, " invalid column count returned by MySQL\n");
  exit(0);
}

/* Execute the SELECT query */
if (mysql_execute(stmt))
{
  fprintf(stderr, " mysql_execute(), failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Bind the result buffers for all 3 columns before fetching them */

/* INTEGER COLUMN */
bind[0].buffer_type= MYSQL_TYPE_LONG;
bind[0].buffer= (char *)&int_data;
bind[0].is_null= &is_null[0];
bind[0].length= &length[0];

/* STRING COLUMN */
bind[1].buffer_type= MYSQL_TYPE_VAR_STRING;
bind[1].buffer= (char *)str_data;
bind[1].buffer_length= STRING_SIZE;
bind[1].is_null= &is_null[1];
bind[1].length= &length[1];

/* SMALLINT COLUMN */
bind[2].buffer_type= MYSQL_TYPE_SHORT;
bind[2].buffer= (char *)&small_data;
bind[2].is_null= &is_null[2];
bind[2].length= &length[2];

/* TIMESTAMP COLUMN */
bind[3].buffer_type= MYSQL_TYPE_TIMESTAMP;
bind[3].buffer= (char *)&ts;
```

```
bind[3].is_null= &is_null[3];
bind[3].length= &length[3];

/* Bind the result buffers */
if (mysql_bind_result(stmt, bind))
{
  fprintf(stderr, " mysql_bind_result() failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Now buffer all results to client */
if (mysql_stmt_store_result(stmt))
{
  fprintf(stderr, " mysql_stmt_store_result() failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Fetch all rows */
row_count= 0;
fprintf(stdout, "Fetching results ...\n");
while (!mysql_fetch(stmt))
{
  row_count++;
  fprintf(stdout, "  row %d\n", row_count);

  /* column 1 */
  fprintf(stdout, "   column1 (integer)  : ");
  if (is_null[0])
    fprintf(stdout, " NULL\n");
  else
    fprintf(stdout, " %d(%ld)\n", int_data, length[0]);

  /* column 2 */
  fprintf(stdout, "   column2 (string)   : ");
  if (is_null[1])
    fprintf(stdout, " NULL\n");
  else
    fprintf(stdout, " %s(%ld)\n", str_data, length[1]);

  /* column 3 */
  fprintf(stdout, "   column3 (smallint) : ");
  if (is_null[2])
    fprintf(stdout, " NULL\n");
  else
    fprintf(stdout, " %d(%ld)\n", small_data, length[2]);
```

```
  /* column 4 */
  fprintf(stdout, "   column4 (timestamp): ");
  if (is_null[3])
    fprintf(stdout, " NULL\n");
  else
    fprintf(stdout, " %04d-%02d-%02d %02d:%02d:%02d (%ld)\n",
                                              ts.year, ts.month, ts.day,
                                              ts.hour, ts.minute, ts.second,
                                              length[3]);
  fprintf(stdout, "\n");
}

/* Validate rows fetched */
fprintf(stdout, " total rows fetched: %d\n", row_count);
if (row_count != 2)
{
  fprintf(stderr, " MySQL failed to return all rows\n");
  exit(0);
}

/* Free the prepared result metadata */
mysql_free_result(prepare_meta_result);


/* Close the statement */
if (mysql_stmt_close(stmt))
{
  fprintf(stderr, " failed while closing the statement\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}
```

12.1.7.14 mysql_send_long_data()

```
my_bool mysql_send_long_data(MYSQL_STMT *stmt, unsigned int parameter_number,
const char *data, ulong length)
```

Descrição

Permite que um aplicação envie os dados dos parâmetros para o servidor em partes (ou "blocos"). Esta função pode ser chamada várias vezes parar enviar partes de valores de dados binários e caracteres para uma coluna, que deve do tipo `TEXT` ou `BLOB`.

`parameter_number` indica a qual parâmetro o dado é associado. Os parâmetros são numerados começando com 0. `data` é um ponteiro para um buffer contendo dados a serem enviados, e `length` indica a quantidade de bytes no buffer.

Valor Retornado

Zero se os dados são enviados com sucesso para o servidir. Diferente de zero se ocorrer um erro.

Erros

`CR_INVALID_PARAMETER_NO`
: Número de parâmetro inválido

`CR_COMMANDS_OUT_OF_SYNC`
: Os comando foram executados em uma ordem inpropriada.

`CR_OUT_OF_MEMORY`
: Falta de memoria.

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_UNKNOWN_ERROR`
: Um erro desconhecido ocorreu.

Example

O exemplo seguinte demonstra como enviar os dados para um coluna do tipo `TEXT` em blocos. Ele insere o dado "MySQL - The most popular open source database" na coluna `text_column`. A variável `mysql` é considerada como um handle de conexão válido.

```
#define INSERT_QUERY "INSERT INTO test_long_data(text_column) VALUES(?)"

MYSQL_BIND bind[1];
long       length;

if (!mysql_prepare(mysql, INSERT_QUERY, strlen(INSERT_QUERY))
{
  fprintf(stderr, "\n prepare failed");
  fprintf(stderr, "\n %s", mysql_error(mysql));
  exit(0);
}
  memset(bind, 0, sizeof(bind));
  bind[0].buffer_type= MYSQL_TYPE_STRING;
  bind[0].length= &length;
  bind[0].is_null= 0;


  /* Bind the buffers */
  if (mysql_bind_param(stmt, bind))
  {
```

```
  fprintf(stderr, "\n param bind failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
}

/* Supply data in chunks to server */
if (!mysql_send_long_data(stmt,0,"MySQL",5))
{
  fprintf(stderr, "\n send_long_data failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
}

/* Supply the next piece of data */
if (mysql_send_long_data(stmt,0," - The most popular open source database",40))
{
  fprintf(stderr, "\n send_long_data failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
}

/* Now, execute the query */
if (mysql_execute(stmt))
{
  fprintf(stderr, "\n mysql_execute failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
}
```

12.1.7.15 mysql_stmt_close()

```
my_bool mysql_stmt_close(MYSQL_STMT *)
```

Descrição

Fecha a instrução preparada. `mysql_stmt_close()` também desaloca o manipulador de instruções apontado por `stmt`.

Se a instrução atual tiver resultados pendentes ou não lidos, esta função os cancela para que a próxima consulta possa ser executada.

Valor Retornado

Zero se a instrução for liberada com sucesso. Diferente de zero se ocorrer um erro.

Erros

`CR_SERVER_GONE_ERROR`
: O servidor MySQL foi finalizado.

`CR_UNKNOWN_ERROR`
: Ocorreu um erro desconhecido.

Exemplo

Para utilizar `mysql_stmt_close()` consulte o exemplo de Seção 12.1.7.5 [`mysql_execute()`], Página 837.

12.1.7.16 `mysql_stmt_errno()`

```
unsigned int mysql_stmt_errno(MYSQL_STMT *stmt)
```

Descrição

Para a instrução especificada por `stmt`, `mysql_stmt_errno()` retorna o código de erro para a função de instruções da API chamada mais recentemente. Um valor de retorno de zero significa que não ocorreu nenhum erro. Números de mensagens de erro do cliente estão listadas no arquivo cabeçlho '`errmsg.h`' do MySQL. Números de mensagens de erro do servidor estão listado no arquivo '`mysqld_error.h`'. Na distribuição fonte do MySQL você pode encontrar uma lista completa de mensagens de erros e número de erros no arquivo '`Docs/mysqld_error.txt`'. Os códigos de erros do servidor também estão listados em Seção 13.1 [Error-returns], Página 889.

Valor Retornado

Um valor de código de erro. Zero se não ocorreu erro.

Erros

Nenhum

12.1.7.17 `mysql_stmt_error()`

```
const char *mysql_stmt_error(MYSQL_STMT *stmt)
```

Descrição

Para a instrução especificada por `stmt`, `mysql_stmt_error()` retorna uma string terminada em null contendo a mensagem de erro para a função de instrução da API chamada mais recentemente. Um string vazia (`""`) é retornado se não ocorreu nenhum erro. Isto significa que os seguintes comandos são equivalentes:

```
if (mysql_stmt_errno(stmt))
{
  // an error occured
}

if (mysql_stmt_error(stmt)[0])
{
  // an error occured
}
```

A linguagem da mensagem de erro do cliente pode ser alterada recompilando a biblioteca cliente do MySQL. Atualmente você pode escolher mensagem de erros em diversas linguagens.

Valor Retornado

Um string contendo a descrição do erro. Uma string vazia se não ocorrer erros.

Erros

Nenhum

12.1.7.18 `mysql_stmt_sqlstate()`

```
const char *mysql_stmt_sqlstate(MYSQL_STMT *stmt)
```

Descrição

Para a intrução especificada por `stmt`, `mysql_stmt_sqlstate()`, retorna uma string terminada em null contendo o código de erro SQLSTATE para função API de instruções preparadas mais recentemente chamada que tenha obtido sucesso ou falhado. O código de erro consiste de cinco caracteres. `"00000"` significa "sem erros". Os valores são especificados pelo ANSI SQL e ODBC. Para uma lista de valores possíveis, veja Seção 13.1 [Error-returns], Página 889.

Note que nem todos os erros já estão mapeados para SQLSTATE. O valor `"HY000"` (erro geral) é usado para erros não mapeados.

Valores Retornados

Uma string terminada em null contendo o código de erro SQLSTATE.

12.1.8 Tratando a Execução de Múltiplas Consultas na API C

A partir da versão 4.1, o MySQL suporta a execução de multiplas instruções especificadas em uma única string de consulta. Para utilizá-lo com uma dada conexão, você deve especificar a

opção `CLIENT_MULTI_STATEMENTS` no parâmetro do `mysql_real_connect()` quando abrir a conexão. Você também pode configurá-la para uma conexão chamando `mysql_set_server_option(MYSQL_OPTION_MULTI_STATEMENTS_ON)`

Por padrão `mysql_query()` ou `mysql_real_query()` retornam apenas o status da primeira consulta e o status das consultas subsequentes podem ser processados usando `mysql_more_results()` e `mysql_next_result()`.

```
/* Connect to server with option CLIENT_MULTI_STATEMENTS */
mysql_real_connect(..., CLIENT_MULTI_STATEMENTS);

/* Now execute multiple queries */
mysql_query(mysql,"DROP TABLE IF EXISTS test_table;\
                   CREATE TABLE test_table(id INT);\
                   INSERT INTO test_table VALUES(10);\
                   UPDATE test_table SET id=20 WHERE id=10;\
                   SELECT * FROM test_table;\
                   DROP TABLE test_table";
do
{
  /* Process all results */
  ...
  printf("total affected rows: %lld", mysql_affected_rows(mysql));
  ...
    if (!(result= mysql_store_result(mysql)))
      {
   printf(stderr, "Got fatal error processing query\n");
           exit(1);
      }
    process_result_set(result); /* client function */

  mysql_free_result(mysql);
}while (!mysql_more_results(mysql))
```

12.1.9 Manipulando Valores de Data e Hora na API C

O novo protocolo binário disponível no MySQL 4.1 e acima lhe permite enviar e receber dados de hora e data (`DATE`, `TIME`, `DATETIME` e `TIMESTAMP`) utilizando a estrutura `MYSQL_TIME`. Os membros desta estrutura estão em Seção 12.1.5 [C API Prepared statement datatypes], Página 828.

Para enviar um valor de dado temporal, você cria uma instrução preparada com `mysql_prepare()`. Então, antes de chamar `mysql_execute()` para executar a instrução, use o seguinte procedimento para configurar cada parâmetro temporal:

1. Na estrutura `MYSQL_BIND` associado com o valor do dado, configure o membro `buffer_type` para o tipo que indique qual tipo de valor temporal você está enviando. Para

valores DATE, TIME, DATETIME, ou TIMESTAMP configure buffer_type para MYSQL_TYPE_DATE, MYSQL_TYPE_TIME, MYSQL_TYPE_DATETIME, ou MYSQL_TYPE_TIMESTAMP repectivamente.

2. Configure o membro buffer da estrutura MYSQL_BIND com o endereço da estrutura MYSQL_TIME na qual você passrá o valor temporal.
3. Preencha os membros da estrutura MYSQL_TIME que são apropriadas para o tipo de valor temporal que você está passando.

Use mysql_bind_param() para ligar os dados do parâmetro a instrução. Então chame mysql_execute().

Para recuperar valores temporais, o procedimento é similar, exceto pelo fato de que você configura o membro buffer_type com o valor que você espera receber e o membro buffer com o endereço de uma estrutura MYSQL_TIME na qual o valor retornado deve ser colocado. Use mysql_bind_results() para ligar o buffer a instrução depois da chamada de mysql_execute() e antes de buscar os resultados.

Aqui está um exemplo simples que insere dados DATE, TIME e TIMESTAMP. A variável mysql é considerada como um handle de conexão válido.

```
MYSQL_TIME  ts;
MYSQL_BIND  bind[3];
MYSQL_STMT  *stmt;

  strmov(query, "INSERT INTO test_table(date_field, time_field,
                                        timestamp_field) VALUES(?,?,?");

  stmt= mysql_prepare(mysql, query, strlen(query)));

  /* define a entrada do buffer com 3 parâmetros */
  bind[0].buffer_type= MYSQL_TYPE_DATE;
  bind[0].buffer= (char *)&ts;
  bind[0].is_null= 0;
  bind[0].length= 0;
  ..
  bind[1]= bind[2]= bind[0];
  ..

  mysql_bind_param(stmt, bind);

  /* fornece os dados a serme enviados na estrutura ts */
  ts.year= 2002;
  ts.month= 02;
  ts.day= 03;

  ts.hour= 10;
  ts.minute= 45;
  ts.second= 20;
```

```
mysql_execute(stmt);
..
```

12.1.10 Descrição das Funções de Threads da API C

Você precisa utilizar as seguintes funções quando quiser criar um cliente em uma thread. Veja Seção 12.1.14 [Clientes em threads], Página 863.

12.1.10.1 `my_init()`

`void my_init(void)`

Descrição

Esta função precisa ser chamada uma vez pelo programa antes de se chamar qualquer função do MySQL. Ela inicializa algumas varáveis globais que o MySQL precisa. se você está usando uma biblioteca cliente de thread segura, também será feita uma chamada a `mysql_thread_init()` para esta thread.

Ela é chamada automaticamente por `mysql_init()`, `mysql_server_init()` e `mysql_connect()`.

Valor Retornado

Nenhum

12.1.10.2 `mysql_thread_init()`

`my_bool mysql_thread_init(void)`

Descrição

Esta função preisa aser chamada para cada thread criada para inicializar variáveis específicas de threads.

Ela é automaticamente chamada por `my_init()` e `mysql_connect()`.

Valor Retornado

Zero se obtver sucesso. Diferente de zero se ocorrer um erro.

12.1.10.3 `mysql_thread_end()`

`void mysql_thread_end(void)`

Descrição

Esta função precisa ser chamada antes da chamada de `pthread_exit()` para liberar a memória alocada por `mysql_thread_init()`.

Note que a função **não é chamada automaticamente** pela biblioteca cliente. Deve ser chamada explicitamente para evitar perda de memória.

Valor Retornado

Nenhum.

12.1.10.4 `mysql_thread_safe()`

```
unsigned int mysql_thread_safe(void)
```

Descrição

Esta função indica se o cliente é compilado como uma thread segura.

Valor Retornado

1 se o cliente possui thread segura, 0 em outro caso.

12.1.11 Descrição das Funções do Servidor Embutido da API C

Você deve utilizar as seguints funções se você quiser permitir que a sua aplicação seja ligada a biblicoteca de servidor MySQL embutido. Veja Seção 12.1.15 [libmysqld], Página 864.

Se o programa é ligado com `-lmysqlclient` em vez de `-lmysqld`, estas funções não farão nada. Isto torna possível escolher entre usar o servidor MySQL embutido e um servidor stand-alone sem modificar nenhum código.

12.1.11.1 `mysql_server_init()`

```
int mysql_server_init(int argc, char **argv, char **groups)
```

Descrição

Esta função **deve** ser chamada uma vez no program usando o servidor embutido antes de se chamar qualquer iutra função do MySQL. Ela inicia o servidor e inicializa qualquer subsistema (`mysys`, `InnoDB`, etc.) que o servidor utilize. Se esta função não for chamada, o programa irá falhar. Se você estiver usando o pacote DBUG que vem com o MySQL, você deve chamar esta função depois de ter chamado `MY_INIT()`.

Os argumentos `argc` e `argv` são análogos ao argumentos para o `main()`. O primeiro elemento de `argv` é ignorado (ele contém normalmente, o nome do programa). por conveniência, `argc`

pode ser `0` (zero) se não houver argumentos de linha de comando para o servidor. `mysql_server_init()` faz uma copia dos argumentos, assim é seguro destruir `argv` ou `groups` depois da chamada.

A lista de strings terminadas em `NULL` em `groups` seleciona qual grupo no arquivo de opções será ativado. Veja Seção 4.1.2 [Arquivos de opções], Página 216. Por conveniência, `groups` deve ser `NULL`, caso no qual os grupos `[server]` d `[emedded]` estarão ativos.

Exemplo

```
#include <mysql.h>
#include <stdlib.h>

static char *server_args[] = {
  "this_program",       /* this string is not used */
  "--datadir=.",
  "--key_buffer_size=32M"
};
static char *server_groups[] = {
  "embedded",
  "server",
  "this_program_SERVER",
  (char *)NULL
};

int main(void) {
  mysql_server_init(sizeof(server_args) / sizeof(char *),
                    server_args, server_groups);

  /* Use any MySQL API functions here */

  mysql_server_end();

  return EXIT_SUCCESS;
}
```

Valor Retornado

0 se okay, 1 se ocorrer um erro.

12.1.11.2 `mysql_server_end()`

`void mysql_server_end(void)`

Descrição

Esta função **deve** ser chamada no programa depois de todas outra funções MySQL. Ela finaliza o srvidor embutido.

Valor Retornado

Nenhum.

12.1.12 Dúvidas e problemas comuns ao utilzar a API C

12.1.12.1 Porque Algumas Vezes `mysql_store_result()` Retorna `NULL` Após `mysql_query()` Returnar com Sucesso?

É possível para `mysql_store_result()` retornar `NULL` seguida de uma chamda com sucesso ao `mysql_query()`. Quando isto acontece, significa que uma da seguintes condições ocorreu:

- Existe um falha no `malloc()` (por exemplo, se o resultado for muito grande).
- Os dados não podem ser lidos (ocorreu um erro na conexão).
- A consulta não retornou dados (por exemplo, ela era um `INSERT`, `UPDATE`, ou `DELETE`).

Você sempre pode verificar se a instrução devia produzir um resultado não vazio chamando `mysql_field_count()`. Se `mysql_field_count()` retornar zero, o resultado está vazio e a última consulta era uma instrução que não devia retorbar valor (por exemplo, um `INSERT` ou um `DELETE`). Se `mysql_field_count()` retorna um valor diferente se zero, a instrução devia ter produzido um resultado não vazio. Veja a descrição da função `mysql_field_count()` para um exemplo.

Você pode testar um erro chamando `mysql_error()` ou `mysql_errno()`.

12.1.12.2 Que Resultados Posso Onbetr de uma Consulta?

Sobre o resultado restornado de uma consulta, você pode obter as seguintes informaçãoes:

- `mysql_affected_rows()` retorna o número de registros afetados pela última consulta ao se fazer uma `INSERT`, `UPDATE`, ou `DELETE`. Uma exceção é que se for utilizado `DELETE` sem uma cláusula `WHERE`, a tabela é recriada vazia, o que é mais rápido! Neste caso, `mysql_affected_rows()` retorna zero para o número de registros afetados.
- `mysql_num_rows()` retorna o número de registros em um resultado. Com `mysql_store_result()`, `mysql_num_rows()` pode ser chamado assim que `mysql_store_result()` retornar. Com `mysql_use_result()`, `mysql_num_rows()` só pode ser chamado depois de ter buscado todos os registros com `mysql_fetch_row()`.
- `mysql_insert_id()` retorna o ID gerado pela última consulta que inseriu um registro em uma tabela com índice `AUTO_INCREMENT`. Veja Seção 12.1.3.32 [`mysql_insert_id()`], Página 803.
- Algumas consultas (`LOAD DATA INFILE ...`, `INSERT INTO ... SELECT ...`, `UPDATE`) retornam informações adcionais. O resultado é retornado por `mysql_info()`. Veja a descrição de `mysql_info()` para o formato da string que ela returnou. `mysql_info()` retorna um ponteiro `NULL` se não houver informações adicionais.

12.1.12.3 Como Posso Obter a ID Única para a Última Linha Inserida?

Se você inserir um registro em uma tabela contendo uma coluna que tiver o atributo `AUTO_INCREMENT`, você pode obter o ID gerado mais recentemente chamando a função `mysql_insert_id()`.

Você também pode recuperar o ID utilizando a função `LAST_INSERT_ID()` em uma string de consulta que foi passada a `mysql_query()`.

Você pode verificar se um índice `AUTO_INCREMENT` é usado executando o seguinte código. Ele também verifica se a consulta era um `INSERT` com um índice `AUTO_INCREMENT`:

```
if (mysql_error(&mysql)[0] == 0 &&
    mysql_num_fields(result) == 0 &&
    mysql_insert_id(&mysql) != 0)
{
    used_id = mysql_insert_id(&mysql);
}
```

O ID gerado mais recentemente é mantido no servidor em uma base por conexão. Ele não será alterado por outro cliente. Ele não será alterado mesmo se você atualizar outra coluna `AUTO_INCREMENT` com um valor não mágico (isto é, um valor que não é `NULL` e nem `0`).

Se você quiser utilizar o ID que foi gerado por uma tabela e inserido em uma segunda tabela, você ode utilizar instruções SQL como esta:

```
INSERT INTO foo (auto,text)
    VALUES(NULL,'text');              # gera ID inserindo NULL
INSERT INTO foo2 (id,text)
    VALUES(LAST_INSERT_ID(),'text');  # usa ID na segunda tabela
```

12.1.12.4 Problemas com Ligação na API C

Ar ligar com a API C, os segintes error podem ocorrem em alguns sistemas:

```
gcc -g -o client test.o -L/usr/local/lib/mysql -lmysqlclient -lsocket -lnsl

Undefined       first referenced
 symbol         in file
floor           /usr/local/lib/mysql/libmysqlclient.a(password.o)
ld: fatal: Symbol referencing errors. No output written to client
```

Se isto acontecer em seu sistema, você deve incluir a biblioteca math adiconando `-lm` ao fim da linha de compilação/ligação.

12.1.13 Construindo Programas Clientes

Se você compilar clientes MySQL escritos por você mesmo ou obtido de terceiros, else devem ser ligados utilizando a opção `-lmysqlclient -lz` no comando de ligação. Você também pode prcisar de especificar uma opção `-L` para dizer ao ligado onde encntrar a biblioteca. Por exemplo, se a biblioteca é instalada em '`/usr/local/mysql/lib`', use `-L/usr/local/mysql/lib -lmysqlclient -lz` no comando de ligação.

Para clientes que utilizam arquivos de cabeçalho do MySQL, pode ser necessário especificar a opção `-I` ao compilá-los, (por exemplo, `-I/usr/local/mysql/include`), assim o compilador pode encontrar o arquivo de cabeçalho.

Para o mostrado acima de forma simples no Unix, fornecemos o script `mysql_config` para você. Veja Seção 4.9.11 [mysql_config], Página 370.

Você pode utilizá-lo para compila o cliente MySQL como a seguir:

```
CFG=/usr/local/mysql/bin/mysql_config
sh -c "gcc -o progname `$CFG --cflags` progname.c `$CFG --libs`"
```

`sh -c` é necessário para fazer com que a sheel não trate a saída de `mysql_config` como uma palavra.

12.1.14 Como Fazer um Cliente em Threads

A biblioteca cliente é quase segura com threads. O maior problema é que a subrotinas em '`net.c`' que leem dos sockets não são seguras a interrupções. Isto foi feito pesando que você pudesse desejar ter o seu próprio alarme que possa quebrar uma longa leitura no servidor. Se você instalar manipuladores de interrupção para a interrupção `SIGPIPE`, o manipulador socket deve ser segura com threads.

Nos binários antigos que distribuímos em nosso web site (`http://www.mysql.com/`), as bibliotecas clientes não estão normalmente compiladas com a opção de segurança com thread (os binários são complados com segurança com thread por padrão). Distribuições binárias mais novas devem ter uma biblioteca normal e uma segura com threads.

Para termos um cliente em threads onde você pode interromper o cliente a partir de outras threads a definir tempo limites ao falar com o servidor MySQL, você deve utilizar as bibliotecas `-lmysys`, `-lmystrings`, e `-ldbug` e o código `net_serv.o` que o servidor utiliza.

Se você não precisar de insterrupções ou de tempos limites, você pode apenas compilar um biblioteca cliente (`mysqlclient_r`) segura com threads e utilizá-las. Veja Seção 12.1 [API C MySQL], Página 775. Neste caso você não precisa se preocupar com o arquivo objeto `net_serv.o` ou outras bibliotecas MySQL.

Quando usar um cliente em thread e você quiser utilizar tempos limite e interrupções, você pode ter um grande uso das rotinas no arquivo '`thr_alarm.c`'. Se você estiver utilizando rotinas da biblioteca `mysys`, a única coisa que você deve lembrar é de chamar primeiro `my_init()`! Veja Seção 12.1.10 [Funções Threads do C], Página 858.

Todas as funções com excessão de `mysql_real_connect()` são seguras com thread por padrão. As anotações seguintes descrevem como compilar uma biblioteca cliente segura com thread e utilizá-la de maneira segura. (As anotações abaixo para `mysql_real_connect()` na verdade se aplicam também a `mysql_connect()`, mas como `mysql_connect()` está obsoleto, você deve utilizar `mysql_real_connect()`.)

Para tornar `mysql_real_connect()` seguro com thread, você deve recompilar a biblioteca cliente com este comando:

```
shell> ./configure --enable-thread-safe-client
```

Isto irá criar uma biblioteca cliente `libmysqlclient_r`. (Assumindo que o seu SO tenha a função `gethostbyname_r()` segura com thread). Esta biblioteca é segura com thread por conexão. Você pode deixar duas threads compartilharem a mesma conexão com os seguintes cuidados:

- Duas threads não podem enviar uma consaulta ao servidor MySQL ao mesmo tempo na mesma conexão. Em particular, você deve assegurar que entre um `mysql_query()` e `mysql_store_result()` nenhuma outra thread está usando a mesma conexão.
- Várias threads podem acessár resultados diferentes que são recuperados com `mysql_store_result()`.
- Se você utilizar `mysql_use_result`, você terá que assegurar que nenhuma outra thread está usando a mesma conexão até que o resultado seja fechado. No entanto, é melhor para clientes em threads que compartilham a mesma conexão utilizar `mysql_store_result()`.
- Se você quiser utilizar múltiplas threads na mesma conexão, você deve ter uma trava mutex na combinação das chamadas `mysql_query()` e `mysql_store_result()`. Uma vez que `mysql_store_result()` esteja pronto, a trva pode ser liberada e outras threads podem utilizar a mesma conexão.
- Se você programa com threads POSIX, você pode utilizar `pthread_mutex_lock()` e `pthread_mutex_unlock()` para estabelecer e liberar uma trava mutex.

Você precisa saber o seguinte se você tiver uma thread que estiver chamando funções MySQL que não criaram a conexão ao banco de dados MySQL:

Quando você chamar `mysql_init()` ou `mysql_connect()`, MySQL irá criar um variável especica da thread para a thread que é utilizada pela bibklioteca de depuração (entre outra coisas).

Se você chamar uma função MySQL, antes da thread chamar `mysql_init()` ou `mysql_connect()`, a thread não terá as variáveis específicas de thread necessárias alocadas e você acabará finalizando com uma descarga de memória mais cedo ou mais tarde.

Para fazer que as coisas funcionem suavemente você tem que fazer o seguinte:

1. Chama `my_init()` no início do seu programa se for chamar qualquer outra função MySQL antes de chamar `mysql_real_connect()`.
2. Chame `mysql_thread_init()` no manipulador de thread antes de chamar qualquer outra função MySQL.
3. Na thread, chame `mysql_thread_end()` antes de chamar `pthread_exit()`. Isto irá liberar a memória usada pelas variáveis específicas da thread do MySQL.

Você pode obter alguns erros devido a símbolos indefinidos ao ligar seu cliente com `libmysqlclient_r`. Na maioria dos casos isto ocorre por não estar incluída a biblioteca de threads na linha de ligação/compilação.

12.1.15 libmysqld, a Biblioteca do Servidor Embutido MySQL

12.1.15.1 Visão Geral da Biblioteca do Servidor MySQL Embutido

A biblioteca do servidor MySQL embutido torna possível executar um servidor MySQL com todos os recursos dentro de uma aplicação cliente. Os principais benefícios são o aumento de velocidade e o gerenciamento mais simples de aplicações embutidas.

A biblioteca do servidor embutido é baseada na versão cliente/servidor do MySQL, que é escrita em C/C++. Consequentemente, o servidor embutido também é escrito em C/C++. Não há nenhum servidor embutido disponível em outra linguagem.

A API é idêntica para a versão embutida do MySQL e a versão cliente/servidor. Para alterar uma aplicação em thread antiga para utilizar a biblioteca embutida, você normalmente só precisa adicionar chamadas as seguintes funções:

Função	Quando chamar
`mysql_server_init()`	Deve ser chamada antes de qualquer outra função MySQL, de preferência no inicio da função `main()`.
`mysql_server_end()`	Deve ser chamada antes da saída do programa.
`mysql_thread_init()`	Deve ser chamada em cada thread que você criar que acessará o MySQL.
`mysql_thread_end()`	Deve ser chamada antes de se chamar `pthread_exit()`

Você deve ligar seu código com '`libmysqld.a`' em vez de '`libmysqlclient.a`'.

As funções acima `mysql_server_xxx` também estão incluídas em '`libmysqlclient.a`' para permitir a troca entre a versão embutida e a clienete/servidor apenas ligando sua aplicação na biblioteca certa. Veja Seção 12.1.11.1 [`mysql_server_init()`], Página 859.

12.1.15.2 Compilando Programas com `libmysqld`

Para obter uma biblioteca `libmysqld` você deve configurar o MySQL com a opção `--with-embedded-server`.

Quando você liga o seu programa com `libmysqld`, você também deve incluir a biblioteca específica do sistema `pthread` e algumas bibliotecas que o servidor MySQL utiliza. Você pode conseguir a lista completa de bibliotecas executando `mysql_config --libmysqld-libs`.

Os parâmetros corretos para compilar e ligar um programa em thread devem ser usados, mesmo se você não chamar nenhuma função thread diretamente em seu código.

12.1.15.3 Restrições no Uso de um Servidor MySQL Embutido

O servidor embutido tem as seguintes limitações:

- Não tem suporte a tabelas ISAM. (Isto é feito para tornar a biblioteca menor)
- No possui funções definidas pelo usuário (UDF).
- Não ratreia pilha em caso de descarga de memória.
- Sem suporte a RAID interno. (Normalmente não é necessário já que a maioria dos SO possui suporte a arquivos grandes).
- Você pode configurá-lo como servidor ou master (sem replicação).
- Você não pode conectar ao servidor embutido de um processo externo com sockets ou TCP/IP.

Algumas desta limitações podem ser alteradas editando o arquivo '`mysql_embed.h`' e re-compilando o MySQL.

12.1.15.4 Usando Arquivo de Opções com o Servidor Embutido

O descrito abaixo é o modo recomendado de utilizar arquivos de opções para facilitar a troca entre uma aplicação cliente/servidor é uma onde o MySQL está embutido. Veja Seção 4.1.2 [Arquivos de opção], Página 216.

- Coloque as seções comuns na seção `[server]`. Ela será lida por ambas as versões do MySQL.
- Coloque a opcões específicas do cliente/servidor na seção `[mysqld]`.
- Coloque as opções específicas do MySQL embutido na seção `[embedded]`.
- Coloque as opções específicas da aplicação na seção `[ApplicationName_SERVER]`.

12.1.15.5 Itens a Fazer no Servidor Embutido (TODO)

- Estamos fornecendo opções para deixar de fora algumas partes do MySQL para tornar a biblioteca menor.
- Ainda há muita otimização de velocidade a se fazer.
- O erros são escritos no `stderr`. Adicionaremos uma opção para especificar um nome de arquivo para eles.
- Temos que alterar o `InnoDB` para não ser tão descritivo quando usado em um servidor embutido.

12.1.15.6 Um Exemplo Simples de Servidor Embutido

Este programa e makefile exemplo devem funcionar sem nenhuma alteração em um sistema Linux ou FreeBSD. Para outros sistemas operacionais, pequenas mudanças serão necessárias. Este exemplo é feito para dar detalhes suficientes para enteder o problema, sem a desordem que é uma parte necessária de uma aplicação real.

Para experimentar o exemplo, crie um diretório '`test_libmysqld`' no mesmo nível que o diretório fonte do mysql-4.0. Salve o fonte '`test_libmysqld.c`' e o '`GNUmakefile`' no diretório e execute GNU '`make`' de dentro do diretório '`test_libmysqld`'.

'`test_libmysqld.c`'

```
/*
 * A simple example client, using the embedded MySQL server library
 */

#include <mysql.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>

MYSQL *db_connect(const char *dbname);
void db_disconnect(MYSQL *db);
void db_do_query(MYSQL *db, const char *query);
```

```
const char *server_groups[] = {
  "test_libmysqld_SERVER", "embedded", "server", NULL
};

int
main(int argc, char **argv)
{
  MYSQL *one, *two;

  /* mysql_server_init() devve ser chamado antes de qualquer
   * função mysql.
   *
   * Você pode usar mysql_server_init(0, NULL, NULL), e iniciar
   * o servidor usando os grupos = {
   *   "server", "embedded", NULL
   *  }.
   *
   * Em seu arquivo $HOME/.my.cnf file, você provavelmente deseja colocar:

[test_libmysqld_SERVER]
language = /path/to/source/of/mysql/sql/share/english

   * É claro que você poderia modifcar argc e argv antes de passá-los
   * a esta função.  Ou poderá criar novos do modo que preferir. Mas
   * todos os argumentos em argv (exceto argv[0], que é o nome do
   * programa) devem ser opções válidas para o servidor MySQL.
   *
   * Se você ligar este cliente em um biblioteca mysqlclient
   * normal, esta função não fará nada.
   */
  mysql_server_init(argc, argv, (char **)server_groups);

  one = db_connect("test");
  two = db_connect(NULL);

  db_do_query(one, "SHOW TABLE STATUS");
  db_do_query(two, "SHOW DATABASES");

  mysql_close(two);
  mysql_close(one);

  /* Isto deve ser chamado depois de todas outras funções mysql*/
  mysql_server_end();

  exit(EXIT_SUCCESS);
}
```

```
static void
die(MYSQL *db, char *fmt, ...)
{
  va_list ap;
  va_start(ap, fmt);
  vfprintf(stderr, fmt, ap);
  va_end(ap);
  (void)putc('\n', stderr);
  if (db)
    db_disconnect(db);
  exit(EXIT_FAILURE);
}

MYSQL *
db_connect(const char *dbname)
{
  MYSQL *db = mysql_init(NULL);
  if (!db)
    die(db, "mysql_init failed: no memory");
  /*
   * Certifique-se que o cliente e o servidor utilizam grupos diferentes.
   * Isto é critico pois o servidor não aceitará as opções do
   * cliente e vice versa.
   */
  mysql_options(db, MYSQL_READ_DEFAULT_GROUP, "test_libmysqld_CLIENT");
  if (!mysql_real_connect(db, NULL, NULL, NULL, dbname, 0, NULL, 0))
    die(db, "mysql_real_connect failed: %s", mysql_error(db));

  return db;
}

void
db_disconnect(MYSQL *db)
{
  mysql_close(db);
}

void
db_do_query(MYSQL *db, const char *query)
{
  if (mysql_query(db, query) != 0)
    goto err;

  if (mysql_field_count(db) > 0)
  {
    MYSQL_RES   *res;
    MYSQL_ROW    row, end_row;
```

```
    int num_fields;

    if (!(res = mysql_store_result(db)))
      goto err;
    num_fields = mysql_num_fields(res);
    while ((row = mysql_fetch_row(res)))
    {
      (void)fputs(">> ", stdout);
      for (end_row = row + num_fields; row < end_row; ++row)
        (void)printf("%s\t", row ? (char*)*row : "NULL");
      (void)fputc('\n', stdout);
    }
    (void)fputc('\n', stdout);
    mysql_free_result(res);
  }
  else
    (void)printf("Affected rows: %lld\n", mysql_affected_rows(db));


  return;

err:
  die(db, "db_do_query failed: %s [%s]", mysql_error(db), query);
}
```

'GNUmakefile'

```
# This assumes the MySQL software is installed in /usr/local/mysql
inc      := /usr/local/mysql/include/mysql
lib      := /usr/local/mysql/lib

# If you have not installed the MySQL software yet, try this instead
#inc      := $(HOME)/mysql-4.0/include
#lib      := $(HOME)/mysql-4.0/libmysqld

CC       := gcc
CPPFLAGS := -I$(inc) -D_THREAD_SAFE -D_REENTRANT
CFLAGS   := -g -W -Wall
LDFLAGS  := -static
# You can change -lmysqld to -lmysqlclient to use the
# client/server library
LDLIBS    = -L$(lib) -lmysqld -lz -lm -lcrypt

ifneq (,$(shell grep FreeBSD /COPYRIGHT 2>/dev/null))
# FreeBSD
LDFLAGS += -pthread
else
# Assume Linux
```

```
LDLIBS += -lpthread
endif

# This works for simple one-file test programs
sources := $(wildcard *.c)
objects := $(patsubst %c,%o,$(sources))
targets := $(basename $(sources))

all: $(targets)

clean:
rm -f $(targets) $(objects) *.core
```

12.1.15.7 Licensiando o Servidor Embutido

O código fonte do MySQL é coberto pela licenção GNU `GPL` (veja Apêndice H [Licença GPL], Página 1096). Um resultado disto é que qualquer programa que incluam, na ligação com `libmysqld`, o código fonte do MySQL deve ser distribuído como software livre. (sob uma licença compatível com a `GPL`).

Nós encorajamos a todos a promover o software livre distribuindo o código sob a `GPL` ou uma licença compatível. Para aqueles que não puderem fazê-lo, outra opção é comprar um licença comercial para o código MySQL da MySQL AB. Para maiores detalhes, consulte Seção 1.4.3 [MySQL licenses], Página 18.

12.2 Suporte ODBC ao MySQL

O MySQL fornece suporte para ODBC através do programa `MyODBC`. Este capítulo lhe ensinará como instalar o `MyODBC`, e como usá-lo. Aqui, você também encontrará uma lista de programas comuns que são conhecidos por funcionar com `MyODBC`.

12.2.1 Como Instalar o MyODBC

`MyODBC` 2.50 é um driver de nível 0 (com recursos de nível 1 e 2) e especificação de ODBC 2.50 de 32 bits para conectar um programa ODBC ao MySQL. `MyODBC` funciona nos Windows 9x/Me/NT/2000/XP e na maioria da plataformas Unix. `MyODBC` 3.51 é uma versão melhorada com nível 1 (core API completo + recursos de nível 2) e especificação de ODBC 3.5x.

`MyODBC` é `Open Source`, e você pode encontrar a versão mais nova em `http://www.mysql.com/downloads/api-myodbc.html`. Note que a versão 2.50.x utiliza a licença `LGPL`, enquanto a versão 3.51.x utiliza a licença `GPL`.

Se vcê tiver problemas com o `MyODBC` e seu programa também funciona com OLEDB, você deve experimentar o driver OLEDB.

Normalmente você só precisa instalar o `MyODBC` em másquinas Windows. Você só precisará de `MyODBC` para Unix se tiver um programa como ColdFusion que roda em máquinas Unix e utilizam ODBC para conectar ao banco de dados.

Se você quiser instalar `MyODBC` em um Unix, você precisará de um gerenciador ODBC. `MyODBC` funciona com a maioria dos gerenciadores ODBC para Unix.

Para instalar `MyODBC` no Windows, você deve baixar o arquivo `MyODBC` '`.zip`' apropriado, descompactá-lo com `WinZip` ou algum programa parecido e executar o arquivo '`SETUP.EXE`'.

No Windows/NT/XP você pode obter o seguinte erro ao tentar instalar o `MyODBC`:

```
An error occurred while copying C:\WINDOWS\SYSTEM\MFC30.DLL. Restart
Windows and try installing again (before running any applications which
use ODBC)
```

O problema neste caso é que algum outro progrma está utilizando ODBC e pela forma que como o Windows é feito, você pode, neste caso, não estar apto a instalar um novo driver ODBC com programa de instação do ODBC da Microsoft. Neste caso você pode continuar selecionando `Ignore` para copiar o resto dos aerquivos ODBC e a instalação final deve funcionar. Se não funcionar, a solução é reinicializar seu computador em "modo seguro" (Escolhendo-o ao pressionar F8 assim que seu computador iniciar o Windows durante a reinicialização), instalar `MyODBC`, e reiniciar em modo normal.

- Para conectar a uma máquina Unix de uma máquina Windows, com uma aplicação ODBC (uma que não tenha suporte nativo as MySQL), você deve primeiro instalar `MyODBC` em uma máquina Windows.
- O usuário é máquina Windows devem ter privilégios para acessar o servidor MySQL na máquina Unix. Isto pode ser feito om o comando `GRANT`. Veja Seção 4.4.1 [`GRANT`], Página 254.
- Você deve criar uma entrada ODBC DSN como a seguir:
  - Abra o painel de controle na máquina Windows.
  - Dê um duplo clique no ícone Fonte de Dados ODBC 32-bit.
  - Clique na seção Usuário DSN.
  - Clique no botão Adicionar.
  - Selecione MySQL na tela Criar Nova Fonte de Dados e clique no botão Finalizar.
  - A tela de configuração padrão do Driver MySQL é mostrada. Veja Seção 12.2.2 [Administrador ODBC], Página 871.
- Agora inicie a sua aplicação e selcione o driver ODBC com o DSN que você especificou no adminitrador ODBC.

Verifique se há outra opção de configuração na tela do MySQL (trace, não pergunta ao conectar, etc) que você possa tentar se ocorrerem problemas.

12.2.2 Como Preencher os Vários Campos no Programa de Administração do ODBC

Existem três maneiras possíveis de especificar o nome de servidor no Windows95:

- Use o endereço IP no servidor.
- Adicione um arquivo '`\Windows\lmhosts`' com a seguinte informação:

  ```
  ip nome_maquina
  ```

  Por exemplo:

```
194.216.84.21 meu_nome_maquina
```

- Configure o PC para utilizar DNS.

Exemplo de como preencher a `configuração do ODBC`.

```
Windows DSN name:   test
Description:        This is my test database
MySQL Database:     test
Server:             194.216.84.21
User:               monty
Password:           my_password
Port:
```

O valor para o campo `Windows DSN name` é qualquer nome que seja único em sua configuração ODBC Windows.

Você não precisa especificar valores para os campos `Server`, `User`, `Password`, ou `Port` ina tela de configuração do ODBC. No entanto, se você o fizer, os valores serão utilizados como padrão posteriormente ao se tentar fazer uma nova conexão. Você tem a opção de alterar os valores neste momento.

Se o número da porta não é dado, a porta padrão (3306) é utilizada.

Se você especificar a opção `Read options from C:\my.cnf`, os grupos `client` e `odbc` serão lidos do arquivo '`C:\my.cnf`'. Você pode utilizar todas as opções que são úteis a `mysql_options()`. Veja Seção 12.1.3.40 [`mysql_options()`], Página 809.

12.2.3 Parâmetros de Conexão do MyODBC

Pode-se especificar os seguintes parâmetros para `MyODBC` na seção `[Servername]` de um arquivo '`ODBC.INI`' ou através do argumento `InConnectionString` na chamada `SQLDriverConnect()`.

Parâmetro	Valor padrão	Comentário
user	ODBC (on Windows)	O nome do usuário usado para se conectar ao MySQL.
server	localhost	O nme de máquina do servidor MySQL.
database		O banco de dados padrão.
option	0	Um inteiro com o qual você pode especificar como o `MyODBC` deve tarbalhar. Veja abaixo.
port	3306	A porta TCP/IP usada se o servidor (`server`) não for `localhost`.
stmt		Uma instrução que será executada ao conectar ao `MySQL`.
password		A senha para a combinação servidor(`server`)-usuário(`user`).
socket		O socket ou pipe Windows para se conectar.

O argumento option é usado para dizer ao `MyODBC` que o cliente não é 100% compatível com ODBC. No Windows, o parâmetro option normalmente é definido mudando as diferentes opções na tela de conexão mas também podem ser definidas no argumento option. As seguintes opções estão listadas na mesma ordem em que aparecem na tela de conexão do `MyODBC`:

Bit	Descrição

1	O cliente não pode aceitar que `MyODBC` retorne a largura real de uma coluna.
2	O clinete nã pode aceitr que MySQL retorne o valor real de colunas afetadas. Se este parâmetro for definido o MySQL retornará 'registros encontrados'. É necessáriop o MySQL 3.21.14 ou posterior para funcionar.
4	Faz um log de depuração em c:\myodbc.log. É o mesmo que colocar `MYSQL_DEBUG=d:t:O,c::\myodbc.log` no '`AUTOEXEC.BAT`'
8	Não define nenhum limite de pacote para resultados e parâmetros.
16	Não faz perguntas mesmo se o driver quisesse.
32	Simula um driver ODBC 1.0 em alguns contextos.
64	Ignora o uso do nome de banco de dados 'bancodedados.tabela.coluna'
128	Força o usa de cursores de gerenciadores ODBC (experimental).
256	Disabilita o uso de busca estendida (experimental).
512	Completa campos CHAR para tamanho de coluna cheias.
1024	SQLDescribeCol() retrnará nome de colunas totalmente qualificados.
2048	Usa o protocolo cliente/servidor comprimido.
4096	Diz ao seridor para ignorar espaços após nome de funções e antes de `'('` (necessário para PowerBuilder). Torna todos os nomes de funções palavras-chaves!
8192	Conecta com named pipes ao servidor `mysqld` executando no NT.
16384	Altera colunas LONGLONG para colunas INT (algumas aplicações não podem tratar LONGLONG).
32768	Retorna 'user' como Table_qualifier e Table_owner para SQLTables (experimental)
65536	Lê paraâmetros dos grupos `client` e `odbc` no '`my.cnf`'
131072	Adiciona algumas verificações extras de segurança (não deve ser necessário, mas...)

Se você quiser ter muitas opções, você deve somar os parâmetros acima! Por exemplo, definir a opção como 12 (4+8) lhe permite debugar sem limite de pacotes.

O '`MYODBC.DLL`' padrão é compilado para um rendimento otimizado. Se você quiser depurar o `MyODBC` (por exemplo, habiliatr o trace), você deve utilizar '`MYODBCD.DLL`'. Para instalar este arquivo copie '`MYODBCD.DLL`' sobre o arquivo '`MYODBC.DLL`' instalado.

12.2.4 Como Relatar Problemas com o MyODBC

`MyODBC` tem sido testado com Access, Admndemo.exe, C++-Builder, Borland Builder 4, Equipe de Desenvovimento Centura (formalmente Gupta SQL/Windows), ColdFusion (em Solaris e NT com svc pack 5), Crystal Reports, DataJunction, Delphi, ERwin, Excel, iHTML, FileMaker Pro, FoxPro, Notes 4.5/4.6, SBSS, Perl DBD-ODBC, Paradox, Powerbuilder, Powerdesigner 32 bit, VC++, e Visual Basic.

Se você souber de qualquer outra aplicação que funcione com `MyODBC`, envie-nos email para lista de email `odbc` do MySQL sobre isto! Veja Seção 1.7.1.1 [Mailing-list], Página 33.

Com alguns programas você pode obter um erro como este: `Another user has modifies the record that you have modified`. Na maioria dos casos ele pode ser resolvido fazendo o especificado abaixo:

- Adicione um chave primária para a tabela se já não houver uma.
- Adicione uma coluan timestamp se já não existir uma.

- Só utilize campos double float. Alguns programa podem falhar quando comparam floats simples.

Se o exposto acima não ajudar, você deve fazer um arquivo de rastreamento do `MyODBC` e tentar encontrar o que está dando errado.

12.2.5 Programas que Funcionam com MyODBC

A maioria dos programas devem funcionar com `MyODBC`, mas para cada um dos listados aqui, nós mesmos testamos e recebemos confirmação de alguns usuários que eles funcionam:

Programa **Comentário**

Access Para fazer o Access funcionar:

- Se você estiver usando Access 2000, você deve obter e instalar o Microsoft MDAC (`Microsoft Data Access Components`) mais recente (versão 2.6 ou acima) em `http://www.microsoft.com/data/`. Ele irá consertar o o seguinte bug no Access: quando você exporta dados para o MySQL, os nomes de tabelas e colunas não são especificados. Outro modo de contornar este bug é atualizar para MyODBC Versão 2.50.33 e MySQL Versão 3.23.x, que juntos fornecem um modo de contornar este erro!

  Você também deve obter e instalar Microsoft Jet 4.0 Service Pack 5 (SP5) que pode ser encontrado em `http://support.microsoft.com/support/kb/articles/Q239/1/14.ASP`. Isto irá corrigir alguns casos onde colunas são marcadas como `#deletadas#` no Access.

  Note que se você estiver usando o MySQL Versão 3.22, você deve aplicar o patch MDAC e utilizar MyODBC 2.50.32 ou 2.50.34 e acima para contornar este problema.
- Para todas as versões do Access, você deve habilitar a opção do MyODBC `Return matching rows`. Para Access 2.0, você também deve habilitar `Simulate ODBC 1.0`.
- Você deve ter um timestamp em todas as tabelas que você deseja atualizar. Para maior portabilidade `TIMESTAMP(14)` ou apenas `TIMESTAMP` é recomendado no lugar de outras variações de `TIMESTAMP(X)`.
- Você deve ter uma chave primáia na tabela. Se não, registros novos ou atualizados podem aparecer como `#DELETED#`.
- Só use campos `DOUBLE` float. O Access falaha ao comparar com campos single floats. Os sintomas normais são que registros novos ou atualizados podem aparecer como `#DELETED#` ou que você não possa encontarar ou atualizar registros.
- Se você estiver ligando uma tabela com MyODBC, que tem `BIGINT` como uma de suas colunas, o resultado será mostrado como `#DELETED`. A solução para contornar este problema é:
  - Tenha um ou mais colunas modelos com `TIMESTAMP` como o tipo de dados, de preferência `TIMESTAMP(14)`.
  - Verifique `'Change BIGINT columns to INT'` na ciaxa diálogo de opções de conexão no Admiistrador DSN ODBC

- Delete o link a tabela e o recrie.

  Ele ainda mostra o registro anterior como `#DELETADO#`, mas novos registros adicionados/atualizados serão mostrados apropriadamente.

- Se você ainda obter o erro `Another user has changed your data` depois de adicionar uma coluna `TIMESTAMP`, o seguinte truque pode lhe ajudar:

  Nào utilize visualizar planilha de dados da `tabela`. Crie um formulario com os campos que você quer, e use visulizar planilha de dados de `formulário`. Você deve definir a propriedade `DefaultValue` para a coluna `TIMESTAMP` com `NOW()`. Esta pode ser uma boa idéia para oculta a coluna `TIMESTAMP` da visualização para que seus usuários não fiquem confusos.

- Em alguns casos, o Access pode gerar consultas SQL inválidas que o MySQL não entende. Você pode arrumar isto selecionando `"Query|SQLSpecific|Pass-Through"` no menu do Access.
- No NT o Access irá mostrar colunas `BLOB` como `OLE OBJECTS`. Se você quiser colunas `MEMO`, você deve alterar a coluna para `TEXT` com `ALTER TABLE`.
- O Access não pode sempre tratar colunas `DATE` apropriadamente. Se você tiver um problema com isto, mude as colunas para `DATETIME`.
- Se você tiver no Acces um coluna definida como `BYTE`, o Access tentará exportá-la como `TINYINT` em vez de `TINYINT UNSIGNED`. Isto lhe tará problemas se você tiver valores > 127 na coluna!

ADO

Quando você está codificando com a API ADO e `MyODBC` você precisa ter atenção com algumas propriedades padrões que não são suportadas pelo servidor MySQL. Por exemplo, usando `CursorLocation Property` como `adUseServer` retornará de `RecordCount Property` um resultado de -1. Para ter o valor correto, você precisa definir esta propriedade a `adUseClient`, como mostrado no código VB abaixo:

```
Dim myconn As New ADODB.Connection
Dim myrs As New Recordset
Dim mySQL As String
Dim myrows As Long

myconn.Open "DSN=MyODBCsample"
mySQL = "SELECT * from user"
myrs.Source = mySQL
Set myrs.ActiveConnection = myconn
myrs.CursorLocation = adUseClient
myrs.Open
myrows = myrs.RecordCount

myrs.Close
myconn.Close
```

Outro modo de contornar o problea é utilizar uma instrução `SELECT COUNT(*)` para uma consulta parecida para obter a contagem de registros correta.

Active server pages (ASP)
: Você deve usar a opção `Return matching rows`.

Aplicações BDE
: Para faze-las funcionar, você deve definir os seguintes parâmetros: `Don't optimize column widths` e `Return matching rows`.

Borland Builder 4
: Qaundo você inicia uma consulta, você pode utilizar a propriedade `Active` ou utilizar o método `Open`. Note que `Active` irá iniciar automaticamente executando uma consulta `SELECT * FROM ...` que pode não ser algo bom se suas tabelas forem grandes.

ColdFusion (No Unix)
: A seguinte informações é tirada da documentação do ColdFusion:

  Utilize a seguinte informação para configurar o ColdFusion Server para Linux para utilizar o driver unixODBC driver com `MyODBC` para fonte de dados MySQL. Allaire verificou que o `MyODBC` Versão 2.50.26 funciona com MySQL Versão 3.22.27 e ColdFusion para Linux. (Quqlquer versão mais nova também deve funcionar.) Você pode fazer o download do `MyODBC` em `http://www.mysql.com/downloads/api-myodbc.html`

  ColdFusion Versão 4.5.1 lhe permite utilizar o Administrador ColdFusion para adicionar a fonte de dados MySQL. No entanto o driver não está incluído com o ColdFusion Versão 4.5.1. Antes que o driver MySQL aparecer na lista drop-down de fonte de dados ODBC, você deve construir e copiar o driver `MyODBC` para '`/opt/coldfusion/lib/libmyodbc.so`'.

  O diretório Contrib contém o programa '`mydsn-xxx.zip`' que lhe permite construir e remover o arquivo de registro DSN para o driver MyODBC em aplicações Coldfusion.

DataJunction
: Você tem que alterá-lo para uma saída `VARCHAR` em vez de `ENUM`, ja que ele exporta o último de uma maneira que cause um grief no MySQL.

Excel
: Funciona. Algumas dicas:

  - Se você tiver problema com datas, tente selecioná-las como strings utilizando a função `CONCAT()`. Por exemplo:

    ```
    select CONCAT(rise_time), CONCAT(set_time)
        from sunrise_sunset;
    ```

    Valores retornados deste modo como strings devem ser reconhecidos corretamente como valores time pelo Excel97.

    O propósito de `CONCAT()` neste exemplo é enganar o ODBC fazendo-o pensar que a coluna é do "tipo string". Sem o `CONCAT()`, ODBC sabe que a coluna é do tipo time e o Excel não entende isto.

    Note que este é um bug do Excel, pois ele converte automaticamente uma string para um time. Isto seria ótimo se a fonte fosse um arquivo texto, mas se torna um erro quando a fonte e uma conexão ODBC que relata tipos exatos para cada coluna.

Word

Para recuperar os dados do MySQL para documentos Word/Excel, você precisa utilizar o driver `MyODBC` e a ajuda do Add-in Microsoft Query.

Por exemplo, crie um bd com uma tabela contendo 2 colunas de texto:

- Insira registros utilizando a ferramente cliente de linha de comando `mysql`.
- Crie um arquivo DSN usando o gerenciador ODBC, '`my`', por exemplo, para o bd acima.
- Abra o aplicativo Word.
- Crie um novo documento vazio.
- Utilizando a barra de ferramentas chamada Banco de Dados, pressione o botão insira banco de dados.
- Pressione o botão Obter Dados.
- No moemnto certo, pressione o botão Ms Query na tela Obter Dados.
- No Ms Query crie uma Nova Fonte de Dados utilizando o arquivo DSN my.
- Selecione a nova consulta.
- Selecione as colunas que você deseja.
- Crie um filtro de desejar.
- Faça um Ordenação se quiser.
- Selecione Enviar Dados para o Microsoft Word.
- Clique em Finalizar.
- Clique em Inserir dados e selecione os registros.
- Clique OK e você verá os registros no seu documento Word.

odbcadmin

Pograma teste para ODBC.

Delphi

Você deve utilizar o BDE Versão 3.2 ou mais atual. Veja a opção i `Don't optimize column width` ao conectar no MySQL.

Aqui está um código de Delphi potencialmente útil que configura uma entrada ODBC e uma entrada BDE para para `MyODBC` (a entrada BDE exige de um Editor de Alias BDE que é gratuito em um site Delphi Super Page. (Obrigado a Bryan Brunton `bryan@flesherfab.com` por isto):

```
fReg:= TRegistry.Create;
  fReg.OpenKey('\Software\ODBC\ODBC.INI\DocumentsFab', True);
  fReg.WriteString('Database', 'Documents');
  fReg.WriteString('Description', ' ');
  fReg.WriteString('Driver', 'C:\WINNT\System32\myodbc.dll');
  fReg.WriteString('Flag', '1');
  fReg.WriteString('Password', '');
  fReg.WriteString('Port', ' ');
  fReg.WriteString('Server', 'xmark');
  fReg.WriteString('User', 'winuser');
  fReg.OpenKey('\Software\ODBC\ODBC.INI\ODBC Data Sources', True);
```

```
fReg.WriteString('DocumentsFab', 'MySQL');
fReg.CloseKey;
fReg.Free;

Memo1.Lines.Add('DATABASE NAME=');
Memo1.Lines.Add('USER NAME=');
Memo1.Lines.Add('ODBC DSN=DocumentsFab');
Memo1.Lines.Add('OPEN MODE=READ/WRITE');
Memo1.Lines.Add('BATCH COUNT=200');
Memo1.Lines.Add('LANGDRIVER=');
Memo1.Lines.Add('MAX ROWS=-1');
Memo1.Lines.Add('SCHEMA CACHE DIR=');
Memo1.Lines.Add('SCHEMA CACHE SIZE=8');
Memo1.Lines.Add('SCHEMA CACHE TIME=-1');
Memo1.Lines.Add('SQLPASSTHRU MODE=SHARED AUTOCOMMIT');
Memo1.Lines.Add('SQLQRYMODE=');
Memo1.Lines.Add('ENABLE SCHEMA CACHE=FALSE');
Memo1.Lines.Add('ENABLE BCD=FALSE');
Memo1.Lines.Add('ROWSET SIZE=20');
Memo1.Lines.Add('BLOBS TO CACHE=64');
Memo1.Lines.Add('BLOB SIZE=32');

AliasEditor.Add('DocumentsFab','MySQL',Memo1.Lines);
```

C++ Builder

Testado com BDE versão 3.0. O único problema conhecido é que quando o esquema de tabelas é alterado, os campos da consulta não são atualizados. O BDE, no entanto, parece não reconhecer chaves primárias, apenas o índice PRIMARY, mas isto será um problema.

Vision Você deve utilizar a opcão `Return matching rows`.

Visual Basic

Para estar apto para habilitar uma tabela, você deve definir uma chave primária para a tabela.

Visual Basic com ADO não pode manipular inteiros grandes. Isto significa que algumas consultas como `SHOW PROCESSLIST` não irão funcionar apropriadamente. A correção é deifinir a opção `OPTION=16384` na string de conexão ODBC ou configurar a opção `Change BIGINT columns to INT` na tela de conexão do MyODBC. Você pode desejar definir a opção `Return matching rows`.

VisualInterDev

Se você obtem o erro `[Microsoft][ODBC Driver Manager] Driver does not support this parameter` a razão pode ser que você tem um `BIGINT` em seu resultado. Tente definir a opção `Change BIGINT columns to INT` na tela de conexão do MyODBC.

Visual Objects

Você deve utilizar a opção `Don't optimize column widths`.

12.2.6 Como Obter o Valor de uma Coluna AUTO_INCREMENT no ODBC

Um problema comum é como obter o valor de um ID gerado automaticamente em um INSERT. Com ODBC, você pode fazer algo assim (considerando que auto é um campo AUTO_INCREMENT):

```
INSERT INTO foo (auto,text) VALUES(NULL,'text');
SELECT LAST_INSERT_ID();
```

Ou, se você estiver prestes a insrir o ID em outra tabela, você pode fazer assim:

```
INSERT INTO foo (auto,text) VALUES(NULL,'text');
INSERT INTO foo2 (id,text) VALUES(LAST_INSERT_ID(),'text');
```

Veja Seção 12.1.12.3 [Obtendo um ID único], Página 862.

Para o benefício de alguns aplicativos ODBC (pelo menos Delphi e Access), a seguinte consulta pode ser utilizada para encontrar um registro récem inserido:

```
SELECT * FROM nome_tabela WHERE auto IS NULL;
```

12.2.7 Relatando Problemas com MyODBC

Se você encontrar dificuldades com MyODBC, você deve iniciar fazendo um arquivo log pelo gerenciador ODBC (o log obtido ao requisitar logs do ODBCADMIN) e um log MyODBC.

Para obter um log MyODBC, você precisa fazer o seguinte:

1. Esteja certo de que você está utilizando 'myodbcd.dll' e não 'myodbc.dll'. O modo mais fácil de se fazer isto é obter 'myodbcd.dll' da distribuição MyODBC e copiá-la sobre o 'myodbc.dll', o qual estará, provavelmente, em seu diretório 'C:\Windows\system32' ou 'C:\winnt\system32'.

   Note que você provavelmente deseja restaurar o myodbc.dll antigo ao finalizar o teste, já que ele é bem mais rpido que 'myodbcd.dll'.

2. Marque a opção 'Trace MyODBC' na tela de conexão/configuração do MyODBC. O logo será escrito no arquivo 'C:\myodbc.log'.

   Se a opção de ratreamento não for lembrada quando você retornar a tela acima, significa que você não está utilizando o driver myodbcd.dll (veja o item acima).

3. Inicie sua aplicação e tente fazê-la falhar.

Verifique o arquivo de rastreamento do MyODBC, para saber o que pode estar errado. Viocê deve estar apto a encontrar as consultas executadas buscando após a string >mysql_real_query no arquivo 'myodbc.log'.

Você também devev tentar duplicar as consultas no monitor mysql ou admndemo para descobrir se o erro é do MyODBC ou do MySQL.

Se você encontar algo errado, envie-nos somente os registros relevantes (max 40 registros) para a lista de email odbc do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33. Por favor, nunca envie todo o arquivo log do MyODBC ou ODBC!

Se você não puder encontrar o que está errado, a última opção é fazer um arquivo (tar ou zip) que contenha um arquivo de rastreamento do MyODBC, o arquivo log do ODBC, e um arquivo README que explique o problema. Você pode enviá-lo para

`ftp://support.mysql.com/pub/mysql/secret/`. Somente nós da MySQL AB teremos acesso ao arquivo que você enviar, a seremos bem discretos com os dados!

Se você pode criar um programa que também mostre este problema, nos envie ele também.

Se o programa funciona com algum outro servidor MySQL, você deve fazer um arquivo de log do MyODBC onde você faz exatamente a mesma coisa no ouuto servidor SQL.

Lembre-se que quanto mais informações você nos fornecer, mais satisfatória será a solução encontrada para o problema!

12.3 Conectividade Java (JDBC) ao MySQL

Existem 2 drivers JDBC suportados pelo MySQL:

- `MySQL Connector/J` do MySQL AB, implementado 100% em Java nativo. Este produto era formalmente conhecido como o driver `mm.mysql`. Você pode fazer o doenload do `MySQL Connector/J` em `http://www.mysql.com/products/connector-j/`.
- O driver Resin JDBC, que pode ser encontrado em `http://www.caucho.com/projects/jdbc-mysql/i`

Para informação, consulte qualquer documentação JDBC, além das documentação dos proprietários de cada driver para recursos específicos do MySQL.

12.4 API PHP do MySQL

PHP é uma linguagem script do lado do servidor embutida em HTML que pode ser usada para criar páginas web dinâmicas. Ele contém suporte para acesso a diversos banco de dados, incluindo o MySQL. PHP pode ser executado como um programa separado ou compilado como um módulo para uso com o servidro web Apache.

A distribuição e documentação está disponível no web site PHP (`http://www.php.net/`).

12.4.1 Problemas Comuns com MySQL e PHP

- Error: `"Maximum Execution Time Exceeded"` Este é um limite do PHP; vá até o arquivo '`php3.ini`' e defina o tempo máximo de excução para algo maior que 30 segundos, de acordo com a necessidade. Também não é uma má idéia dobrar a ram permitida por script para 16 MB em vez de 8 MB.
- Error: `"Fatal error: Call to unsupported or undefined function mysql_connect() in ..."` Isto significa que sua versão do PHP não é compilada com suporte ao MySQL. Você também pode compilar um módulo MySQL dinâmico e carregá-lo no PHP ou recompilar o PHP com suporte ao MySQL. Isto é descrito em detalhes no manual PHP.
- Error: `"undefined reference to 'uncompress'"` Isto significa que a biblioteca cliente é compilada com suporte a um protocolo cliente/servidor compactado. A correção é adicionar `-lz` por último ao ligar com `-lmysqlclient`.

12.5 API Perl do MySQL

Esta seção documenta a interface Perl `DBI`. A interface anterior era chamada `mysqlperl`. `DBI/DBD` é a intrface Perl recomendada atual;mente, assim `mysqlperl` está obsoleta e não será documentada aqui.

12.5.1 `DBI` com `DBD::mysql`

`DBI` é uma interface genérica para muitos bancos de dados. Isto significa que voccê pode escrever um script que funciona com diferentes mecanismos de banco de dados sem nenhuma mudança. Você precisa de um DataBase Driver - Driver de Banco de Dados (DBD) definido para cada tipo de banco de dados, para o MySQL este driver é chamado `DBD::mysql`.

Para mais informação sobre Perl5 DBI, visite a pagina web do `DBI` e leia a documentação:

```
http://dbi.perl.org/
```

Note que se você quiser usar transações com Perl, você precisa ter o `DBD-mysql` versão 1.2216 ou posterior. Recomendamos usar a versão 2.1022 ou mais nova.

Installation instructions for MySQL Perl support are given in Seção 2.7 [Perl support], Página 164.

Se você tiver o modulo MySQL instalado, você pode achar informação sobre as funcionalidades especificas do MySQL com um dos seguintes comandos:

```
shell> perldoc DBD/mysql
shell> perldoc mysql
```

12.5.2 A interface `DBI`

Métodos e Atributos DBI Portáteis

Método/Atributo	Descrição
`connect`	Estabelece uma conexão ao servidor de banco de dados.
`disconnect`	Disconecta de um servidor de banco de dados.
`prepare`	Prepara uma instrução SQL para ser executada.
`execute`	Executa instruções preparadas.
`do`	Prepara e executa uma instrução SQL.
`quote`	Coloca valores string ou `BLOB` entre aspas para serem inseridos.
`fetchrow_array`	Busca a próxima linha como um vetor de campos.
`fetchrow_arrayref`	Busca a próxima linha como um vetor referência de campos.
`fetchrow_hashref`	Busca a prima linha como uma referência a uma tabela hash.
`fetchall_arrayref`	Busca todos os dados como um vetor de vetor (matriz).
`finish`	Finaliza uma instrução e deixa os recursos do sistema livres.
`rows`	Retorna o número de linhas afetadas.
`data_sources`	Retorna um vetor de banco de dados disponíves ne localhost.

`ChopBlanks`	Controla de o método `fetchrow_*` elimina os espaços em branco.
`NUM_OF_PARAMS`	O número de colchetes em uma instrução preparada.
`NULLABLE`	Quais colunas podem ser `NULL`.
`trace`	Realiza rastreamento para depuração.

Métodos e Atributos específicos do MySQL

Método/Atributos	**Descrição**
`mysql_insertid`	O último valor `AUTO_INCREMENT`.
`is_blob`	Quais colunas são valores `BLOB`.
`is_key`	Quais colunas são chaves.
`is_num`	Quais colunas são numéricas.
`is_pri_key`	Quais colunas são chaves primárias.
`is_not_null`	Quais colunas NÃO PODEM ser `NULL`. Veja `NULLABLE`.
`length`	O tamanho máximo das colunas.
`max_length`	O tamanho máximo das colunas presentes no resultado.
`NAME`	Nomes de colunas.
`NUM_OF_FIELDS`	Número de campos retornados.
`table`	Nome de tabelas no resultado.
`type`	Todos os tipos de colunas

Os métodos Perl são descritos em maiores detalhes nas seções seguintes. Variáveis usadas em métodos que retornam valor tem estes significados:

`$dbh`	Manipulador do Banco de Dados
`$sth`	Manipulador da Instrução
`$rc`	Código de Retorno (geralmente um status)
`$rv`	Valor de Retorno (geralmente um contador de linhas)

Métodos e Atributos DBI Portáteis

`connect($data_source, $username, $password)`

Usa o método `connect` para fazer uma conexão do banco de dados a fonte de dados. O valor `$data_source` deve começar com `DBI:driver_name:`. Exemplo de uso de `connect` com o driver `DBD::mysql`:

```
$dbh = DBI->connect("DBI:mysql:$database", $user, $password);
$dbh = DBI->connect("DBI:mysql:$database:$hostname",
                    $user, $password);
$dbh = DBI->connect("DBI:mysql:$database:$hostname:$port",
                    $user, $password);
```

Se o nome do usuário e/ou senha não são definidos, `DBI` usa os valores das variáveis de anbiente `DBI_USER` e `DBI_PASS`, respctivamente. Se você não especificar um nome de máquina, ele utiliza o padrão `'localhost'`. Se você não especificar um número de porta, ele utiliza a porta padrão do MySQL(3306).

Atá o `Msql-Mysql-modules` Versão 1.2009, o valor `$data_source` permitia alguns modificadores:

`mysql_read_default_file=file_name`
: Lê '`file_name`' como um arquivo de opção. Para informação sobre arquivo de opções veja Seção 4.1.2 [Option files], Página 216.

`mysql_read_default_group=group_name`
: O grupo padrão ao se ler uma arquivo de opções é, normalamente, o grupo `[client]`. Especificando a aopção `mysql_read_default_group`, o grupo padrão se torna o grupo `[group_name]`.

`mysql_compression=1`
: Utiliza comunicação compactada enter o cliente e o servidor (MySQL Versão 3.22.3 ou posterior).

`mysql_socket=/path/to/socket`
: Especifica o caminho do socket Unix que é utilizado para se conectar ao servidor (MySQL Versão 3.21.15 ou posterior).

Múliplos modificadores podem ser dados. Cada um deve ser precedido de ponto e vírgula.

Por exemplo, se você quiser evitar colocar o nome de usuário e senha em um script `DBI`, você pode buscá-los em um arquivo de opção '`~/.my.cnf`' do usuário ao invés de escrever a sua chamada `connect` desta forma:

```
$dbh = DBI->connect("DBI:mysql:$database"
                    . ";mysql_read_default_file=$ENV{HOME}/.my.cnf",
                    $user, $password);
```

Esta chamado irá ler opções difinidas pelo grupo `[client]` no arquivo de opções. Se você quiser fazer a mesma coisa mas utilizar opções especificadas no grupo `[perl]`, você pode fazer:

```
$dbh = DBI->connect("DBI:mysql:$database"
                    . ";mysql_read_default_file=$ENV{HOME}/.my.cnf"
                    . ";mysql_read_default_group=perl",
                    $user, $password);
```

`disconnect`
: O método `disconnect` disconecta o manipulador de banco de dados do banco de dados. Ele é normalmente chamado pouco antes de você sair do programa. Exemplo:

```
$rc = $dbh->disconnect;
```

`prepare($statement)`
: Prepara uma instrução SQL para execução pelo mecanismo de banco de dados e retorna um manipulador de instrução (`$sth`), que você pode utilizar para chamar o método `execute`.

Normalmente você manipula a instrução `SELECT` (e instruções do tipo `SELECT` tais como `SHOW`, `DESCRIBE`, e `EXPLAIN`) através de `prepare` e `execute`. Exemplo:

```
$sth = $dbh->prepare($statement)
    or die "Can't prepare $statement: $dbh->errstr\n";
```

Se voê quiser ler grandes resultados em seu cliente, você pode dizer ao Perl para utilizar `mysql_use_result()` com:

```
my $sth = $dbh->prepare($statement { "mysql_use_result" => 1});
```

`execute` O método `execute` executa um instrução preparada. Para instrução não-`SELECT`, `execute` retorna o número de linha afetadas. Se nenhuma linha foi afetada, `execute` retorna `"0E0"`, que o Perl trata como zero mas considera com true. Se um erro ocorrer, `execute` retorna `undef`. Para instruções `SELECT`, `execute` apenas inicia a consulta SQL no banco de dados; você precisa utilizar um dos métodos de `fetch_*` descritos aqui para recuperar dados. Exemplo:

```
$rv = $sth->execute
          or die "can't execute the query: " . $sth->errstr;
```

`do($statement)`

O método `do` prepara e executa uma instrução SQL e retorna o número linhas afetadas. Se nenhuma lina for afetada, `do` retorna `"0E0"`, que o Perl trata como zero mas considera como true (verdadeiro). Este método é geralmente usado por instruções não-`SELECT` que não podem ser preparadas previamente (devida a limitações do driver) ou que não precisa ser esecutada mais que uma vez (inserts, deletes, etc.). Exemplo:

```
$rv = $dbh->do($statement)
        or die "Can't execute $statement: $dbh- >errstr\n";
```

Geralamente a instrução 'do' é mais rápida (e preferível) que prepare/execute para instruções que não contém parâmetros.

`quote($string)`

O método `quote` é usada para "escapar" qualquer caracter especial contido na string e para adcionar as aspas necessárias na saída. Exemplo:

```
$sql = $dbh->quote($string)
```

`fetchrow_array`

Este mátodo busca a próxima linha de dados e a retorna como um vetor de valores de campo. Exemplo:

```
while(@row = $sth->fetchrow_array) {
        print qw($row[0]\t$row[1]\t$row[2]\n);
}
```

`fetchrow_arrayref`

Este método busca a próxima linha de dados e a retorna como uma referência a um vetor de valores de campos. Exemplo:

```
while($row_ref = $sth->fetchrow_arrayref) {
        print qw($row_ref->[0]\t$row_ref->[1]\t$row_ref->[2]\n);
}
```

`fetchrow_hashref`

Este método busca uma linha de dados e retorna uma referência a uma tabela hash contendo pares nome de campos/valores. Este método não é tão eficiente quanto utilizar referências a vetor como demostrado acima. Exemplo:

```
while($hash_ref = $sth->fetchrow_hashref) {
        print qw($hash_ref->{firstname}\t$hash_ref->{lastname}\t\
                $hash_ref->{title}\n);
```

```
}
```

`fetchall_arrayref`
: Este método é usado para obter todos os dados (linhas) a serem retornados de uma instrução SQL. Ele retorna uma referência a um vetor de referências a vetores para cada linha. Você acessa ou imprime dados utilizando um loop aninhado. Exemplo:

```
my $table = $sth->fetchall_arrayref
                or die "$sth->errstr\n";
my($i, $j);
for $i ( 0 .. $#{$table} ) {
        for $j ( 0 .. $#{$table->[$i]} ) {
                print "$table->[$i][$j]\t";
        }
        print "\n";
}
```

`finish`
: Indica que mais nenhum dado será buscado para este manipulador de instrução. Você chama este método para liberar o manipulador de instrução e qualquer recuros de sistema associado a ele. Exemplo:

```
$rc = $sth->finish;
```

`rows`
: Retorna o número de linhas alteradas (atualiadas, deletadas, etc.) pelo último comando. Ele é normalmente utilizado após uma instrução `execute` não-`SELECT`. Exemplo:

```
$rv = $sth->rows;
```

`NULLABLE`
: Retorna uma referência a um vetor de valores que indicam se colunas podem conter valores `NULL`. Os valores possíveis para cada element do vetor é 0 ou uma string vazia se a coluna não puder ser `NULL`, 1 se puder e 2 se a o estado `NULL` da coluna é desconhecido. Exemplo:

```
$null_possible = $sth->{NULLABLE};
```

`NUM_OF_FIELDS`
: este atributi indica o número de campos retornados pela instrução `SELECT` ou `SHOW FIELDS`. Você pode usá-la para verificar se uma instrução retornou um resultado: Um valor zero indica uma intrução não-`SELECT` como `INSERT`, `DELETE`, ou `UPDATE`. Exemplo:

```
$nr_of_fields = $sth->{NUM_OF_FIELDS};
```

`data_sources($driver_name)`
: Este método retorna um vetor contendo o nome dos bancos de dados disponíveis no servidor MySQL na máquina `'localhost'`. Exemplo:

```
@dbs = DBI->data_sources("mysql");
```

`ChopBlanks`
: Este atributo determina se o método `fetchrow_*` irá apagar espaços em branco no inicio ou no fim dos valores retornados. Exemplo:

```
$sth->{'ChopBlanks'} =1;
```

`trace($trace_level)`
`trace($trace_level, $trace_filename)`

O método `trace` habilita ou disabilita o rastreamento. Quando chamado como um método da classe `DBI`, ele afeta o rastreamento em todos os manipuladores. Quando chamado como um método do manipulador de banco de dados ou de instrução, ele afeta o rastreamento para o manipulador dado (e qualquer filho futuro do manipulador). Definir `$trace_level` com 2 fornece detalhes da informação do rastreamento. Definir `$trace_level` com 0 desabilita o rastreamento. A saída do rastreamento vai para a saída padrão de erros por padrão. Se `$trace_filename` for esecificado, o arquivo é aberto no modo append e a saída para **todos** manipuladores rastreados traced handles é escrita neste arquivo. Exemplo:

```
DBI->trace(2);                # rastreia tudo
DBI->trace(2,"/tmp/dbi.out"); # rastreia tudo para
                              # /tmp/dbi.out
$dth->trace(2);               # rastreia este manipulador de banco de dad
$sth->trace(2);               # rastreia este manipulador de instruções
```

Você também pode habilitar o rastreamento `DBI` configurando a variável de ambiente `DBI_TRACE`. Configurá-la com um valor numérico é o mesmo que chamar `DBI->(value)`. Configurá-la com um caminhao é o mesmo que chamar `DBI->(2,value)`.

Métodos e Atributos Especificos do MySQL

Os métodos mostrados aqui são específicso do MySQL e não são parte do padrão `DBI`. Diversos métodos já estão obsoletos: `is_blob`, `is_key`, `is_num`, `is_pri_key`, `is_not_null`, `length`, `max_length`, e `table`. Quando existir uma alternativa no padrão `DBI`, ela será listada aqui:

`mysql_insertid`

Se você utilizar o recurso `AUTO_INCREMENT` do MySQL, os novos valores auto-increment serão armazenados aqui. Exemplo:

```
$new_id = $sth->{mysql_insertid};
```

Com versões antigas da interface DBI, você pode usar `$sth->{'insertid'}`.

`is_blob` Retorna uma referência a um vetor de valores booleanos; para cada elemento do vetor, um valor TRUE indica que a respectiva coluna é um `BLOB`. Exemplo:

```
$keys = $sth->{is_blob};
```

`is_key` Retorna um referência a um vetor de valores booleanos; para cada elemento do vetor, um valor de TRUE indica que a coluna respectiva é uma chave. Exemplo:

```
$keys = $sth->{is_key};
```

`is_num` Retorna uma referência a um vetor de valores booleanos; para cada elemento do vetor, um valor de TRUE indica que a coluna respectiva contém valores numéricos. Exemplo:

```
$nums = $sth->{is_num};
```

`is_pri_key`
: Retorna uma referência a um vetor de valores booleanos; para cada elemento do vetor, um valor de TRUE indica que a respectiva coluna é uma chave primária. Exemplo:

```
$pri_keys = $sth->{is_pri_key};
```

`is_not_null`
: Retorna uma referência para um vetor de valores booleanos; para cada elemento do vetor, um valor de FALSE indica que esta coluna pode conter valores `NULL` Exemplo:

```
$not_nulls = $sth->{is_not_null};
```

`is_not_null` está obsoleto; é preferível utilizar o atributo `NULLABLE` (descrito acima), porque ele é um padrão DBI.

`length`
`max_length`
: Cada um destes métodos retornam uma referêcia a um vetor com tamanho de colunas. O vetor `length` indica a tamanho máximo que cada coluna pode ter (como declarado na descrição da tabela). O vetor `max_length` indica o tamanho máximo presente atualmente no resultado. Exemplo:

```
$lengths = $sth->{length};
$max_lengths = $sth->{max_length};
```

`NAME` Retorna um referência a um vetor de nomes de colunas. Exemplo:

```
$names = $sth->{NAME};
```

`table` Retorna um referência a um vetor de nomes de tabelas. Exemplo:

```
$tables = $sth->{table};
```

`type` Retorna uma referência a um vetor com tipos de colunas. Exemplo:

```
$types = $sth->{type};
```

12.5.3 Mais Informações `DBI/DBD`

Você pode utilizar o comando `perldoc` para conseguir mais informação sobre `DBI`.

```
perldoc DBI
perldoc DBI::FAQ
perldoc DBD::mysql
```

Voê também pode utilizar as ferramentas `pod2man`, `pod2html`, etc., para traduzir para outro formato.

Você pode encontrar as últimas informações sobre `DBI` na pagina web `DBI`: `http://dbi.perl.org/`.

12.6 API C++ do MySQL

MySQL Connector/C++ (ou `MySQL++`) é a API oficiaL do MySQL para C++. Mais informações podem ser encontradas em `http://www.mysql.com/products/mysql++/`.

12.6.1 Borland C++

Você pode compilar o fonte do MySQL Windows com Borland C++ 5.02. (O fonte Windows só incluem projetos para Microsoft VC++, para Borland C++ você mesmo tem que fazer os arquivos de projetos.)

Um problema conhecido copm o Borland C++ é que ele usa uam estrutura de alinhamento diferente do VC++. Isto significa que você terá problema se você tentar utilizar as bibliotecas `libmysql.dll` padrões (que foi compilado com VC++) com Borland C++. Você pode fazer o seguinte para ebitar este problema.

- Você pode utilizar bibliotecas MySQL estáticas para Borland C++ que você pode encontar em `http://www.mysql.com/downloads/os-win32.html`.
- Só chame `mysql_init()` com `NULL` como um argumento, não uma struct MySQL prealocada.

12.7 API Python do MySQL

`MySQLdb` fornece suporte MySQL para Python, compatível com a API Python DB version 2.0. Ela pode ser encontrada em `http://sourceforge.net/projects/mysql-python/`.

12.8 API Tcl do MySQL

MySQLtcl é uma API simples para aceeso ao servidor de banco de dados MySQL a partir da linguagem de programação Tcl. Ela pode ser encontrada em `http://www.xdobry.de/mysqltcl/`.

12.9 Eiffel Wrapper do MySQL

Eiffel MySQL é uma interface para o servidor de banco de dados MySQL, utilizando a linguagem de programação Eiffel, escrita por Michael Ravits. Ela pode ser encontrada em `http://efsa.sourceforge.net/archive/ravits/mysql.htm`.

13 Tratamento de Erros no MySQL

Este capítulo descreve como o MySQL trata erros.

13.1 Erros Retornados

A seguir estão códigos de erro que podem aparecer quando você chama o MySQL de qualquer lingugem da máquina.

As coluna `Name` e `Error Code` correspondem a definição no arquivo de código fonte do MySQL: '`include/mysqld_error.h`'

A coluna `SQLSTATE` corrsponde a definições no arquivo de código fonte do MySQL: '`include/sql_state.h`'

O código de erro SQLSTATE só aparecerá se você utilizar o MySQL versão 4.1. O código SQLSTATE foi adicionado para compatibilidade com o comportamento de X/Open / ANSI / ODBC.

Um texto sugerido para cada código de erro pode ser encontrado no arquivo de mensagem de erro: '`share/english/errmsg.sys`'

Como atualizações são frequentes, é possível que a fonte acima contenha códigos de erros adicionais.

Name	Error Code	SQLSTATE
ER_HASHCHK	1000	HY000
ER_NISAMCHK	1001	HY000
ER_NO	1002	HY000
ER_YES	1003	HY000
ER_CANT_CREATE_FILE	1004	HY000
ER_CANT_CREATE_TABLE	1005	HY000
ER_CANT_CREATE_DB	1006	HY000
ER_DB_CREATE_EXISTS	1007	HY000
ER_DB_DROP_EXISTS	1008	HY000
ER_DB_DROP_DELETE	1009	HY000
ER_DB_DROP_RMDIR	1010	HY000
ER_CANT_DELETE_FILE	1011	HY000
ER_CANT_FIND_SYSTEM_REC	1012	HY000
ER_CANT_GET_STAT	1013	HY000
ER_CANT_GET_WD	1014	HY000
ER_CANT_LOCK	1015	HY000
ER_CANT_OPEN_FILE	1016	HY000
ER_FILE_NOT_FOUND	1017	HY000
ER_CANT_READ_DIR	1018	HY000
ER_CANT_SET_WD	1019	HY000
ER_CHECKREAD	1020	HY000
ER_DISK_FULL	1021	HY000
ER_DUP_KEY	1022	23000

ER_ERROR_ON_CLOSE	1023	HY000
ER_ERROR_ON_READ	1024	HY000
ER_ERROR_ON_RENAME	1025	HY000
ER_ERROR_ON_WRITE	1026	HY000
ER_FILE_USED	1027	HY000
ER_FILSORT_ABORT	1028	HY000
ER_FORM_NOT_FOUND	1029	HY000
ER_GET_ERRNO	1030	HY000
ER_ILLEGAL_HA	1031	HY000
ER_KEY_NOT_FOUND	1032	HY000
ER_NOT_FORM_FILE	1033	HY000
ER_NOT_KEYFILE	1034	HY000
ER_OLD_KEYFILE	1035	HY000
ER_OPEN_AS_READONLY	1036	HY000
ER_OUTOFMEMORY	1037	HY001
ER_OUT_OF_SORTMEMORY	1038	HY001
ER_UNEXPECTED_EOF	1039	HY000
ER_CON_COUNT_ERROR	1040	08004
ER_OUT_OF_RESOURCES	1041	08004
ER_BAD_HOST_ERROR	1042	08S01
ER_HANDSHAKE_ERROR	1043	08S01
ER_DBACCESS_DENIED_ERROR	1044	42000
ER_ACCESS_DENIED_ERROR	1045	42000
ER_NO_DB_ERROR	1046	42000
ER_UNKNOWN_COM_ERROR	1047	08S01
ER_BAD_NULL_ERROR	1048	23000
ER_BAD_DB_ERROR	1049	42000
ER_TABLE_EXISTS_ERROR	1050	42S01
ER_BAD_TABLE_ERROR	1051	42S02
ER_NON_UNIQ_ERROR	1052	23000
ER_SERVER_SHUTDOWN	1053	08S01
ER_BAD_FIELD_ERROR	1054	42S22
ER_WRONG_FIELD_WITH_GROUP	1055	42000
ER_WRONG_GROUP_FIELD	1056	42000
ER_WRONG_SUM_SELECT	1057	42000
ER_WRONG_VALUE_COUNT	1058	21S01
ER_TOO_LONG_IDENT	1059	42000
ER_DUP_FIELDNAME	1060	42S21
ER_DUP_KEYNAME	1061	42000
ER_DUP_ENTRY	1062	23000
ER_WRONG_FIELD_SPEC	1063	42000
ER_PARSE_ERROR	1064	42000
ER_EMPTY_QUERY	1065	42000
ER_NONUNIQ_TABLE	1066	42000
ER_INVALID_DEFAULT	1067	42000
ER_MULTIPLE_PRI_KEY	1068	42000
ER_TOO_MANY_KEYS	1069	42000

ER_TOO_MANY_KEY_PARTS	1070	42000
ER_TOO_LONG_KEY	1071	42000
ER_KEY_COLUMN_DOES_NOT_EXITS	1072	42000
ER_BLOB_USED_AS_KEY	1073	42000
ER_TOO_BIG_FIELDLENGTH	1074	42000
ER_WRONG_AUTO_KEY	1075	42000
ER_READY	1076	00000
ER_NORMAL_SHUTDOWN	1077	00000
ER_GOT_SIGNAL	1078	00000
ER_SHUTDOWN_COMPLETE	1079	00000
ER_FORCING_CLOSE	1080	08S01
ER_IPSOCK_ERROR	1081	08S01
ER_NO_SUCH_INDEX	1082	42S12
ER_WRONG_FIELD_TERMINATORS	1083	42000
ER_BLOBS_AND_NO_TERMINATED	1084	42000
ER_TEXTFILE_NOT_READABLE	1085	HY000
ER_FILE_EXISTS_ERROR	1086	HY000
ER_LOAD_INFO	1087	HY000
ER_ALTER_INFO	1088	HY000
ER_WRONG_SUB_KEY	1089	HY000
ER_CANT_REMOVE_ALL_FIELDS	1090	42000
ER_CANT_DROP_FIELD_OR_KEY	1091	42000
ER_INSERT_INFO	1092	HY000
ER_UPDATE_TABLE_USED	1093	HY000
ER_NO_SUCH_THREAD	1094	HY000
ER_KILL_DENIED_ERROR	1095	HY000
ER_NO_TABLES_USED	1096	HY000
ER_TOO_BIG_SET	1097	HY000
ER_NO_UNIQUE_LOGFILE	1098	HY000
ER_TABLE_NOT_LOCKED_FOR_WRITE	1099	HY000
ER_TABLE_NOT_LOCKED	1100	HY000
ER_BLOB_CANT_HAVE_DEFAULT	1101	42000
ER_WRONG_DB_NAME	1102	42000
ER_WRONG_TABLE_NAME	1103	42000
ER_TOO_BIG_SELECT	1104	42000
ER_UNKNOWN_ERROR	1105	HY000
ER_UNKNOWN_PROCEDURE	1106	42000
ER_WRONG_PARAMCOUNT_TO_PROCEDURE	1107	42000
ER_WRONG_PARAMETERS_TO_PROCEDURE	1108	HY000
ER_UNKNOWN_TABLE	1109	42S02
ER_FIELD_SPECIFIED_TWICE	1110	42000
ER_INVALID_GROUP_FUNC_USE	1111	42000
ER_UNSUPPORTED_EXTENSION	1112	42000
ER_TABLE_MUST_HAVE_COLUMNS	1113	42000
ER_RECORD_FILE_FULL	1114	HY000
ER_UNKNOWN_CHARACTER_SET	1115	42000
ER_TOO_MANY_TABLES	1116	HY000

ER_TOO_MANY_FIELDS	1117	HY000
ER_TOO_BIG_ROWSIZE	1118	42000
ER_STACK_OVERRUN	1119	HY000
ER_WRONG_OUTER_JOIN	1120	42000
ER_NULL_COLUMN_IN_INDEX	1121	42000
ER_CANT_FIND_UDF	1122	HY000
ER_CANT_INITIALIZE_UDF	1123	HY000
ER_UDF_NO_PATHS	1124	HY000
ER_UDF_EXISTS	1125	HY000
ER_CANT_OPEN_LIBRARY	1126	HY000
ER_CANT_FIND_DL_ENTRY	1127	HY000
ER_FUNCTION_NOT_DEFINED	1128	HY000
ER_HOST_IS_BLOCKED	1129	HY000
ER_HOST_NOT_PRIVILEGED	1130	HY000
ER_PASSWORD_ANONYMOUS_USER	1131	42000
ER_PASSWORD_NOT_ALLOWED	1132	42000
ER_PASSWORD_NO_MATCH	1133	42000
ER_UPDATE_INFO	1134	HY000
ER_CANT_CREATE_THREAD	1135	HY000
ER_WRONG_VALUE_COUNT_ON_ROW	1136	21S01
ER_CANT_REOPEN_TABLE	1137	HY000
ER_INVALID_USE_OF_NULL	1138	42000
ER_REGEXP_ERROR	1139	42000
ER_MIX_OF_GROUP_FUNC_AND_FIELDS	1140	42000
ER_NONEXISTING_GRANT	1141	42000
ER_TABLEACCESS_DENIED_ERROR	1142	42000
ER_COLUMNACCESS_DENIED_ERROR	1143	42000
ER_ILLEGAL_GRANT_FOR_TABLE	1144	42000
ER_GRANT_WRONG_HOST_OR_USER	1145	42000
ER_NO_SUCH_TABLE	1146	42S02
ER_NONEXISTING_TABLE_GRANT	1147	42000
ER_NOT_ALLOWED_COMMAND	1148	42000
ER_SYNTAX_ERROR	1149	42000
ER_DELAYED_CANT_CHANGE_LOCK	1150	HY000
ER_TOO_MANY_DELAYED_THREADS	1151	HY000
ER_ABORTING_CONNECTION	1152	08S01
ER_NET_PACKET_TOO_LARGE	1153	08S01
ER_NET_READ_ERROR_FROM_PIPE	1154	08S01
ER_NET_FCNTL_ERROR	1155	08S01
ER_NET_PACKETS_OUT_OF_ORDER	1156	08S01
ER_NET_UNCOMPRESS_ERROR	1157	08S01
ER_NET_READ_ERROR	1158	08S01
ER_NET_READ_INTERRUPTED	1159	08S01
ER_NET_ERROR_ON_WRITE	1160	08S01
ER_NET_WRITE_INTERRUPTED	1161	08S01
ER_TOO_LONG_STRING	1162	42000
ER_TABLE_CANT_HANDLE_BLOB	1163	42000

ER_TABLE_CANT_HANDLE_AUTO_INCREMENT	1164	42000
ER_DELAYED_INSERT_TABLE_LOCKED	1165	HY000
ER_WRONG_COLUMN_NAME	1166	42000
ER_WRONG_KEY_COLUMN	1167	42000
ER_WRONG_MRG_TABLE	1168	HY000
ER_DUP_UNIQUE	1169	23000
ER_BLOB_KEY_WITHOUT_LENGTH	1170	42000
ER_PRIMARY_CANT_HAVE_NULL	1171	42000
ER_TOO_MANY_ROWS	1172	42000
ER_REQUIRES_PRIMARY_KEY	1173	42000
ER_NO_RAID_COMPILED	1174	HY000
ER_UPDATE_WITHOUT_KEY_IN_SAFE_MODE	1175	HY000
ER_KEY_DOES_NOT_EXITS	1176	HY000
ER_CHECK_NO_SUCH_TABLE	1177	42000
ER_CHECK_NOT_IMPLEMENTED	1178	42000
ER_CANT_DO_THIS_DURING_AN_TRANSACTION	1179	25000
ER_ERROR_DURING_COMMIT	1180	HY000
ER_ERROR_DURING_ROLLBACK	1181	HY000
ER_ERROR_DURING_FLUSH_LOGS	1182	HY000
ER_ERROR_DURING_CHECKPOINT	1183	HY000
ER_NEW_ABORTING_CONNECTION	1184	08S01
ER_DUMP_NOT_IMPLEMENTED	1185	HY000
ER_FLUSH_MASTER_BINLOG_CLOSED	1186	HY000
ER_INDEX_REBUILD	1187	HY000
ER_MASTER	1188	HY000
ER_MASTER_NET_READ	1189	08S01
ER_MASTER_NET_WRITE	1190	08S01
ER_FT_MATCHING_KEY_NOT_FOUND	1191	HY000
ER_LOCK_OR_ACTIVE_TRANSACTION	1192	HY000
ER_UNKNOWN_SYSTEM_VARIABLE	1193	HY000
ER_CRASHED_ON_USAGE	1194	HY000
ER_CRASHED_ON_REPAIR	1195	HY000
ER_WARNING_NOT_COMPLETE_ROLLBACK	1196	HY000
ER_TRANS_CACHE_FULL	1197	HY000
ER_SLAVE_MUST_STOP	1198	HY000
ER_SLAVE_NOT_RUNNING	1199	HY000
ER_BAD_SLAVE	1200	HY000
ER_MASTER_INFO	1201	HY000
ER_SLAVE_THREAD	1202	HY000
ER_TOO_MANY_USER_CONNECTIONS	1203	42000
ER_SET_CONSTANTS_ONLY	1204	HY000
ER_LOCK_WAIT_TIMEOUT	1205	HY000
ER_LOCK_TABLE_FULL	1206	HY000
ER_READ_ONLY_TRANSACTION	1207	25000
ER_DROP_DB_WITH_READ_LOCK	1208	HY000
ER_CREATE_DB_WITH_READ_LOCK	1209	HY000
ER_WRONG_ARGUMENTS	1210	HY000

ER_NO_PERMISSION_TO_CREATE_USER	1211	42000
ER_UNION_TABLES_IN_DIFFERENT_DIR	1212	HY000
ER_LOCK_DEADLOCK	1213	40001
ER_TABLE_CANT_HANDLE_FULLTEXT	1214	HY000
ER_CANNOT_ADD_FOREIGN	1215	HY000
ER_NO_REFERENCED_ROW	1216	23000
ER_ROW_IS_REFERENCED	1217	23000
ER_CONNECT_TO_MASTER	1218	08S01
ER_QUERY_ON_MASTER	1219	HY000
ER_ERROR_WHEN_EXECUTING_COMMAND	1220	HY000
ER_WRONG_USAGE	1221	HY000
ER_WRONG_NUMBER_OF_COLUMNS_IN_SELECT	1222	21000
ER_CANT_UPDATE_WITH_READLOCK	1223	HY000
ER_MIXING_NOT_ALLOWED	1224	HY000
ER_DUP_ARGUMENT	1225	HY000
ER_USER_LIMIT_REACHED	1226	42000
ER_SPECIFIC_ACCESS_DENIED_ERROR	1227	HY000
ER_LOCAL_VARIABLE	1228	HY000
ER_GLOBAL_VARIABLE	1229	HY000
ER_NO_DEFAULT	1230	42000
ER_WRONG_VALUE_FOR_VAR	1231	42000
ER_WRONG_TYPE_FOR_VAR	1232	42000
ER_VAR_CANT_BE_READ	1233	HY000
ER_CANT_USE_OPTION_HERE	1234	42000
ER_NOT_SUPPORTED_YET	1235	42000
ER_MASTER_FATAL_ERROR_READING_BINLOG	1236	HY000
ER_SLAVE_IGNORED_TABLE	1237	HY000
ER_WRONG_FK_DEF	1238	42000
ER_KEY_REF_DO_NOT_MATCH_TABLE_REF	1239	HY000
ER_CARDINALITY_COL	1240	21000
ER_SUBSELECT_NO_1_ROW	1241	21000
ER_UNKNOWN_STMT_HANDLER	1242	HY000
ER_CORRUPT_HELP_DB	1243	HY000
ER_CYCLIC_REFERENCE	1244	HY000
ER_AUTO_CONVERT	1245	HY000
ER_ILLEGAL_REFERENCE	1246	42S22
ER_DERIVED_MUST_HAVE_ALIAS	1247	42000
ER_SELECT_REDUCED	1248	01000
ER_TABLENAME_NOT_ALLOWED_HERE	1249	42000
ER_NOT_SUPPORTED_AUTH_MODE	1250	08004
ER_SPATIAL_CANT_HAVE_NULL	1251	42000
ER_COLLATION_CHARSET_MISMATCH	1252	42000
ER_SLAVE_WAS_RUNNING	1253	HY000
ER_SLAVE_WAS_NOT_RUNNING	1254	HY000
ER_TOO_BIG_FOR_UNCOMPRESS	1255	HY000
ER_ZLIB_Z_MEM_ERROR	1256	HY000
ER_ZLIB_Z_BUF_ERROR	1257	HY000

ER_ZLIB_Z_DATA_ERROR	1258	HY000
ER_CUT_VALUE_GROUP_CONCAT	1259	HY000
ER_WARN_TOO_FEW_RECORDS	1260	01000
ER_WARN_TOO_MANY_RECORDS	1261	01000
ER_WARN_NULL_TO_NOTNULL	1262	01000
ER_WARN_DATA_OUT_OF_RANGE	1263	01000
ER_WARN_DATA_TRUNCATED	1264	01000
ER_WARN_USING_OTHER_HANDLER	1265	01000
ER_CANT_AGGREGATE_COLLATIONS	1266	42000
ER_DROP_USER	1267	42000
ER_REVOKE_GRANTS	1268	42000
ER_CANT_AGGREGATE_3COLLATIONS	1269	42000
ER_CANT_AGGREGATE_NCOLLATIONS	1270	42000
ER_VARIABLE_IS_NOT_STRUCT	1271	HY000
ER_UNKNOWN_COLLATION	1272	HY000
ER_SLAVE_IGNORED_SSL_PARAMS	1273	HY000
ER_SERVER_IS_IN_SECURE_AUTH_MODE	1274	HY000
ER_WARN_FIELD_RESOLVED	1275	HY000
ER_BAD_SLAVE_UNTIL_COND	1276	HY000
ER_MISSING_SKIP_SLAVE	1277	HY000
ER_UNTIL_COND_IGNORED	1278	HY000

14 Estendendo o MySQL

14.1 MySQL Internals

Este capítulo descreve várias coisas que você precisa saber ao trabalhar no código do MySQL. Se você planeja contribuir com o desenvolvimento do MySQL, quiser ter acesso ao código entre versões, ou apenas deseja acompanhar o desenvolvimento, siga as instruções em Seção 2.3.4 [Instalando a árvore de fontes], Página 100. Se você está interessada nos MySQL internals, você também deve se inscrever na nossa lista de emails `internals`. Esta lista é relativamente de baixo tráfico. Para detalhes de como se inscrever, por favor veja Seção 1.7.1.1 [Mailing-list], Página 33. Todos os desenvolvedores na MySQL AB estão na lista `internals` e nós ajudamos outras pessoal que estão trabalhando no código MySQL. Esteja a vontade de utilizar esta tanto para perguntas sobre o código qunato para enviar patches com os auqis você gostaria de contribui no projeto MySQL!

14.1.1 Threads MySQL

O servidor MySQL cria as seguintes threads:

- A thread da conexão TCP/IP trata todas as requisições de conexão e cria uma nova thread dedicada para tratar a autenticação e consulta SQL processada por cada conexão.
- No Windows NT existe um thread que trata named pipe que fazem o mesmo trabalho que as threads da conexão TCP/IP em pedidos de conexão de named pipe.
- A thread de sinal trata todos os sinais. Esta thread também trata normalmente de alarmes e chamadas `process_alarm()` para forçar um tempo limite em conexões que têm estado parados por um tempo grande.
- Se o `mysqld` é compilado com `-DUSE_ALARM_THREAD`, uma thread dedicada que trata dos alarmes é criada. Ela só é utilizadas em alguns sistemas onde há problemas com `sigwait()` ou se deseja utilizar o código `thr_alarm()` em aplicações sem uma thread dedicada para tratar sianis.
- Se é utilizada a opção `--flush_time=#`, uma thread dedicada é criada para descarregar todas as tabelas em um dado intervalo.
- Cada conexão tem a sua própria thread.
- Cada tabela diferente na qual é utilizada `INSERT DELAYED` tem sua própria thread.
- Se você quiser utilizar `--master-host`, uma thread de replicação slave será iniciada para ler e aplicar atualizações do master.

`mysqladmin processlist` mostra apenas a thread da conexão, do `INSERT DELAYED`, e da replicação.

14.1.2 Pacotes de Teste do MySQL

Até pouco tempo, o nosso principal pacote de teste com cobertura total era baseado em dados proprietários de clientes e por esta razão não era disponível publicamente. A única

parte disponível publicamente de nosso processo de teste consistia de um teste `crash-me`, um benchamrk Perl DBI/DBD encontrado no diretório `sql-bench` e testes variadaos localizadaos no diretório `tests`. A falta de um um pacote de teste padronizado disponível publicamente tem criado dificuldade para nosso usuários e para nossos desenvolvedores de fazer teste de regressão no código do MySQL. Para resolver este problema, nós criamos um novo sistema de teste que é incluído nas distribuições fonte e binária do Unix a partir da versão 3.23.29. Os testes podem ser executados no Unix ou no Windows usando um ambiente Cygwin. Eles não podem ser executados em um ambiente Windows nativo.

O conjunto de testes de atual não testa tudo no MySQL, mas deve pegar os bugs mais óbvios no código de processamento SQL, detalhes de SO/biblioteca, e é bem compleo em teste de replicações. Nosso objetivo eventual é ter os testes cobrindo 100% do código. Contibuições para o nosso pacote de teste são benvindas. Você pode desejar contribuir com testes que examinam a funcionalidade critica ao seu sistema, o que irá assegurar que todas as futuras versões do MySQL irão funcionar bem com suas aplicações.

14.1.2.1 Executando o Pacote de Testes do MySQL

O sistema de teste consiste de um interpretador de linguagem de teste (`mysqltest`), um script shell para executar todos os testes (`mysql-test-run`), os casos de teste atual escritos em uma linguagem de teste especial e seus resultados esperados. Para executar o pacote de teste em seu sistema depois de uma construção, digite `make test` ou `mysql-test/mysql-test-run` da raiz do fonte. Se você tiver uma distribuição binária instalada, digite `cd` para a raíz de instalação. (ex. `/usr/local/mysql`), e faça `scripts/mysql-test-run`. Todos os testes devem dar certo. Se não, você deve tentar encontrar o porque e relatar o problema se este é um bug n MySQL. Veja Seção 14.1.2.3 [Relatando bugs no mysqltest], Página 898.

Se você tiver uma cópia de `mysqld` executando ná máquina onde você deseja executar o teste, você não tem de pará-lo, desde que não esteja usando as portas `9306` e `9307`. Se uma destas portas forem tomadas, você deve editar `mysql-test-run` e alterar os valores da porta do master e/ou slave para uma disponível.

Você pode executar um cado de teste individual com `mysql-test/mysql-test-run test_name`.

Se um teste falhar, você de testar executando `mysql-test-run` com a opção `--force` para verificar se nenhum outro teste falhou.

14.1.2.2 Extendendo o Pacote de Teste do MySQL

Você pode utilizar a linguagem `mysqltest` para escrever o seu próprio caso de teste. Infelizmente nós ainda não escrevemos a documentação completa para ela. Você pode, no entanto, olhar os nosso casos de teste atuais e usá-los como um exemplo. O seguintes pontos devem ajudá-lo a começar:

- Os teste estão localizados em `mysql-test/t/*.test`
- Um caso de teste consiste de instruções terminadas em `;` e é similar a entrada do cliente de linha de comando `mysql`. Uma instrução por padrão é uma consulta a ser enviada ao servidor MySQL, a menos que ele seja reconhecido como um comando insterno (ex. `sleep`).

- Todas as consultas que produzem resultadosex., `SELECT`, `SHOW`, `EXPLAIN`, etc., devem ser precedidas com `@/path/to/result/file`. O arquivo deve conter os resultados esperados. Um modo fácil de gerar o arquivo resultante é executar `mysqltest -r < t/test-case-name.test` do diretório `mysql-test`, e então editar o arquivo resultante gerado e, se necessário, ajustá-los a saída esperada. Neste caso, tenha cuidado de não adicionar ou deletar quaisquer caracteres invisíveis - tenha certeza de apenas alterar o texto e/ou adicionar linhas deletadas. Se você tiver que inserir uma linha, esteja certo que os campos são separados com tabulação e que há uma tabulação no final. Você pode querer utilizar `od -c` para ter certeza que seu editor de texto não bagunçou nada durante a edição. Nós, é claro, esperamos que você nunca tenha que editar a saída de `mysqltest -r` já que você só deverá fazê-lo quando encontra um bug.
- Para estar consistente com a nossa configuração, você deve colocar seus arquivos de resultados no diretório `mysql-test/r` e o nomeie como `test_name.result`. Se o teste produzir mais de um resultado, você deve usar `test_name.a.result`, `test_name.b.result`, etc.
- Se uma instrução retornar um erro, você eve espacificar na linha anterior a instrução com `--error error-number`. O número do erro pode ser uma lista de números de erros possíveis separados com `','`.
- Se você estiver escrevendo em teste de replicação, você deve coloca `source include/master-slave.inc;` na primeira linha do arquivo. Para trocar entre master e slave, utilize `connection master;` e `connection slave;`. se você precisar fazer alguma coisa em uma conexão alternativa, você pode fazer `connection master1;` para o master e `connection slave1;` para o slave.
- Se você precisar fazer alguma coisa em um loop, você pode usar algo assim:

```
let $1=1000;
while ($1)
{
 # do your queries here
 dec $1;
}
```

- Para 'dormir' entre consultas, use o comando `sleep`. Ele suporta frações de um segundo, assim você pode fazer `sleep 1.3;`, por exemplo, para dormir 1.3 segundos.
- Para executar o slave com opções adicionais para o seu caso de teste, coloque-os na formato de linha de comando `mysql-test/t/test_name-slave.opt`. Para o master, coloque-os em `mysql-test/t/test_name-master.opt`.
- Se você tiver uma questão sobre o pacote de testes, ou tiver um caso de teste para contribuir, envie um e-mail para lista de email "internals" do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33. Como a lista não aceita anexos, você deve utilizar o ftp para enviar os arquivos relevantes: `ftp://support.mysql.com/pub/mysql/Incoming/`

14.1.2.3 Relatando Bugs no Pacote de Teste do MySQL

Se a sua versão não passar no pacote de teste você deve fazer o seguinte:

- Não envie um relatório de bug antes de ter feito tudo possível para encontrar o que esta errado! Quando o fizer, por favor, utilize o script `mysqlbug` assim podemoster

informações sobre o seu sistema e a versão do `MySQL`. Veja Seção 1.7.1.3 [Relato de Erros], Página 36.

- Esteja certo de inluir a saída de `mysql-test-run`, assim como o conteúdoi de todos os arquivos `.reject` no diretório `mysql-test/r`.
- Se um pacote de teste falhar, verifique se o teste também falha quando executado sozinho:

  ```
  cd mysql-test
  mysql-test-run --local test-name
  ```

  Se falhar, você deve configurar o MySQL com `--with-debug` e executar `mysql-test-run` com a opção `--debug`. Se into também falhar envie o arquivo de rastreamento '`var/tmp/master.trace`' para ftp://support.mysql.com/pub/mysql/secret assim nós podemos examiná-los. Por favor, se lembre de também incluir uma descrição completa do seu sistema, a versão do binário do mysqld e como você o compilou.
- Tente também executar `mysql-test-run` com a opção `--force` para ver se há qualquer outro teste que tenha falhado.
- Se você próprio compilou o MySQL, verifique nosso manual sobre como compilar o MySQL na sua platforma ou, de preferência, use um dos binários que nós compilamos para você no `http://www.mysql.com/downloads/`. Todos os seus binários padrões devem passar no pacote de teste!
- Se você obter um erro, como `Result length mismatch` ou `Result content mismatch`, significa que a saída do teste não é igual a saída esperada. Este pode ser um bug no MySQL ou que o seu versão do mysqld produz resultados um pouco diferentes sobre certas circuntâncias.

  Resultado de testes que falharam são colocados em um arquivo com o mesmo nome base que o arquivo de resultado com a extensão `.reject`. Se o seu caso de teste está falhando, você deve fazer um diff nos dois arquivos. Se você não puder ver como els são diferentes, examine ambos com `od -c` e també verifique os seus tamanhos.
- Se um teste falhar totalmente, você deve verificar os arquivos de log no diretório `mysql-test/var/log` para avisos sobre o que deu errado.
- Se você tiver compilado o MySQL com depuração você pode tentar depurá-lo executando `mysql-test-run` com a opções `--gdb` e/ou `--debug`. Veja Seção E.1.2 [Criando arquivos de rastreamento], Página 1080.

  Se você não tiver compilado o MySQL com depuração você deve, provavelmente, fazê-lo. Apenas especifique a opção `--with-debug` no `configure`! Veja Seção 2.3 [Instalando o Fonte], Página 94.

14.2 Adicionando Novas Funções ao MySQL

Existem dois modos de se adicionar novas funções ao MySQL:

- Você pode adicionar novas funções através da interface de funções definidas por usuários - user-definable function (UDF). Funções definidas por usuários são adicionadas e removidas dinamicamente usando as instruções `CREATE FUNCTION` e `DROP FUNCTION`. Veja Seção 14.2.1 [`CREATE FUNCTION`], Página 900.

- Você pode adicionar as funções como uma função nativa do MySQL. Funções nativas são compiladas no servidor `mysqld` e ficam disponíveis em uma base permanente.

Cada método tem suas vantagens e desvantagens:

- Se você escreve uma função definida pelo usuário, você deve instalar o arquivo objeto no seu servidor. Se você compilou a sua função dentro do servidor você não precisará fazer isto.
- Você pode adicionar UDFs para um distribuição binária MySQL. Funções nativas exigem que você modifique a sua distribuição fonte.
- Se você atualizar a sua ditribuição MySQL, você pode continuar a usar a sua UDF previamente instalada. Para funções nativas, você deve repetir as suas modificações a cada vez que você atualizar.

Seja qual for o método que você utilizou para adicionar novas funções, eles podem ser usados como funções nativas tais como `ABS()` ou `SOUNDEX()`.

14.2.1 Sintaxe `CREATE FUNCTION/DROP FUNCTION`

```
CREATE [AGGREGATE] FUNCTION nome_função RETURNS {STRING|REAL|INTEGER}
       SONAME nome_bibliot_compartilhada

DROP FUNCTION function_name
```

Uma função definida pelo usuário (user-definable function - UDF) é um modo de extender o MySQL com uma nova função que funciona como funções nativas do MySQL tais como `ABS()` e `CONCAT()`.

`AGGREGATE` é uma nova opção do MySQL Versão 3.23. Uma função `AGGREGATE` funciona exatamente como uma função `GROUP` nativa do MySQL como `SUM` ou `COUNT()`.

`CREATE FUNCTION` salva o nome e o tipo da função e o nome da biblioteca compartilhada na tabela do sistema `mysql.func`. Você deve ter privilégios `INSERT` e `DELETE` no banco de dados `mysql` para criar e deletar funções.

Todas as funções ativas são recarregadas a cada vez que o servidor é reiniciado, a menos que você reinicie o `mysqld` com a opção `--skip-grant-tables`. Neste caso, a inicialização de UDF é ignorada e as UDFs estão indisponíveis. (Uma função ativa é aquela que foi carregada com `CREATE FUNCTION` e não foi removida com `DROP FUNCTION`.)

Para instruções sobre como escrever funções denidas por usuários, veja Seção 14.2 [Adding functions], Página 899. Para o mecanisnmo UDF funcionar, as funções dever ser escritas em C ou C++, seu sistema operacional deve suporta carregamento dinâmico e você deve compilar o `mysqld` dinamicamente (e não estaticamente).

Note que para fazer `AGGREGATE` funcioanr, você deve ter uma tabela `mysql.func` que contém a coluna `type`. Se você não tem esta tabela, você deve executar o script `mysql_fix_privilege_tables` para criá-la.

14.2.2 Adicionando Novas Funções Definidas Por Usuário

Para o mecanismo UDF funcionar, as funções devem estar em C ou C++ e o seu sistema operacional deve suporta carregamento dinâmico. A distribuição fonte do MySQL inclui um

arquivo '`sql/udf_example.cc`' que definem 5 novas funções. Consulte este arquivo para ver como a convenção de chamadas UDF funciona.

Para o `mysqld` estar apto a usar funções UDF, você deve configurar o MySQL com `--with-mysqld-ldflags=-rdynamic`. A razão é que para muitas plataformas (incluindo Linux) você pode carregar uma biblioteca (com `dlopen()`) de um programa ligado estaticamente, que você teria se estivesse usando `--with-mysqld-ldflags=-all-static`. Se você quiser usar uma UDF que precisa acessar símbolos do `mysqld` (como o exemplo `metaphone` em '`sql/udf_example.cc`' que usa `default_charset_info`), você deve ligar o programa com `-rdynamic` (veja `man dlopen`).

Se você estiver usando uma versão precompilada do servidor, use o MySQL-Max, que suporta carregamento dinâmico.

Para cada função que você deseja usar nas instruções SQL, você deve definir funções C (ou C++) correspondente. Na discussão abaixo, o nome "xxx" é usado um nome de função exemplo. Para distinguir entre o uso de SQL e C/C++, `XXX()` (maiúscula) indica a chamada da função SQL e `xxx()` (minúscula) indica da chamada da função C/C++.

Aa funções C/C++ que você escreve para implemmentar a interface para `XXX()` são:

`xxx()` (exigido)
: A função principal. É onde o resultado da função é computado. A correspondência entre o tipo SQL e o tipo retornado da sua função C/C++ é mostrada aqui:

Tipo SQL	**Tipo C/C++**
`STRING`	`char *`
`INTEGER`	`long long`
`REAL`	`double`

`xxx_init()` (opcional)
: A função de inicialização para `xxx()`. Ela pode ser usada para:

  - Verifica o número de argumentos para `XXX()`.
  - Verifica se os argumentos são de um tipo exigido ou, alternativamente, diga ao MySQL para converter os argumentos para o tipo desejado quando a função principal é chamada.
  - Aloca a memória exigida pela função principal.
  - Especifica o tamanho máximo do resultado.
  - Especifica (para funções `REAL`) o número máximo de decimais.
  - Especifica se o resultado pode ser `NULL`.

`xxx_deinit()` (opicional)
: A função de finalização para `xxx()`. Ela deve liberar qualquer memória alocada pela função de inicialização.

Quando uma instrução SQL invoka `XXX()`, o MySQL chama a função de inicialização `xxx_init()` para realizar qualquer configuração necessária, tais como verificação de argumentos e alocação de memória. Se `xxx_init()` retorna um erro, a instrução SQL é abortada com uma mensagem e as funções principais e de finalização não são chamadas. Senão, a função principal `xxx()` é chamada uma vez para cada linha. Depois de todas as linhas tiverem

sido processadas, a função de finalização `xxx_deinit()` é chamada, podendo assim realizar qualquer 'limpeza'.

Para funções agregadas (como `SUM()`), você também deve fornecer as seguintes funções:

`xxx_reset()` (exigida)
: Zera a soma e insere um argumento como o valor inicial para um novo grupo.

`xxx_add()` (exigida)
: Adiciona o argumento a soma antiga.

Quando se usa UDF's agregadas o MySQL funciona da seguinte maneira:

1. Chama `xxx_init()` para deixar funções agregadas alocarem a memória necessária para armazenar os resultados.
2. Ordena a tabela de acordo com a expressão `GROUP BY`.
3. Para a primeira linha em um novo grupo, chama a função `xxx_reset()`.
4. Para cada nova linha que pertence ao mesmo grupo, chame a função `xxx_add()`.
5. Quando o grupo muda ou depois da última linha ter sido processada, chame `xxx()` para obter o resultado para o conjunto.
6. Repita 3-5 até que todas as linhas tenham sido processada.
7. Chame `xxx_deinit()` para deixar a UDF liberar a memória alocada.

Todas as funções devem ser seguras com thread (não apenas a função principal, mas também as funções de inicialização e finalização). Isto significa que você não tem permissão para alocar qualquer variável global ou estática que alterou! Se você precisa de memória, você deve alocá-la em `xxx_init()` e liberá-la em `xxx_deinit()`.

14.2.2.1 Sequência de Chamadas UDF para Funções Simples

A função principal deve ser declarada como mostrado aqui. Note que o tipo retornado e os parâmetros diferem, dependendo se você irá declarar a função SQL `XXX()` para retornar `STRING`, `INTEGER`, ou `REAL` na instrução `CREATE FUNCTION`:

Para funções `STRING`:

```
char *xxx(UDF_INIT *initid, UDF_ARGS *args,
          char *result, unsigned long *length,
          char *is_null, char *error);
```

Para funções `INTEGER`:

```
long long xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Para funções `REAL`:

```
double xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

As funções de inicialização e finalização são declaradas desta forma:

```
my_bool xxx_init(UDF_INIT *initid, UDF_ARGS *args, char *message);

void xxx_deinit(UDF_INIT *initid);
```

O parâmetro `initid` é passado para todas as três funções. Ela aponta para uma estrutura `UDF_INIT` que é usada para passar informações entre as funções. Os membros da estrutura `UDF_INIT` são listados abaixo. A função de inicialização deve estar em todos os menbros que desejam ser alterados. (Para utilizar o padrão para um membro, deixe-o inalterado.):

`my_bool maybe_null`
: `xxx_init()` deve definir `maybe_null` com `1` se `xxx()` pode retornar `NULL`. O valor padrão é `1` se qualquer um dos argumentos são declarados como `maybe_null`.

`unsigned int decimals`
: Número de decimais. O valor padrão é o número máximo de deciamis no argumento passado na função principal. (Por exemplo, se a função é passada function is passed `1.34`, `1.345` e `1.3`, o padrão seria 3, pois `1.345` tem 3 decimais.

`unsigned int max_length`
: O tamanho máximo de um resultado string. O valor padrão difere dependendo do tipo de resultado da função. Para funções strings, o padrão é o temanho do maior argumento. Para funções do tipo inteiro, o padrão é 21 digitos. Para funções do tipo real, o padrão é 13 mais o número de decimais indicados por `initid->decimals`. (Para funções numéricas, o tamanho inclui qualquer caracter de sinal ou ponto decimal.)

  Se você quiser retornar um blon, você pode definí-lo com 65K ou 16M; esta memória não é alocada, mas usada para decidir qual tipo de coluna utilizar se houver necessidade dese armazenar dados temporários.

`char *ptr` Um ponteiro que a função pode usar para o seus propósitos. Por exemplo, funções pode usar `initid->ptr` para comunicar memórias alocadas entre funções. Na `xxx_init()`, aloca a memória e a atribui a este ponteiro:

```
initid->ptr = allocated_memory;
```

  Em `xxx()` e `xxx_deinit()`, se refira a `initid->ptr` para usar ou liberar a memória.

14.2.2.2 Sequência de Chamadas UDF para Funções Agregadas

Aqui segue uma descrição das diferentes funções que você precisa definir quando você quer criar uma função UDF agregada.

Note que a seguinte função NÃO é necessária ou usada pelo MySQL 4.1.1. Você ainda pode manter a definição de sua função se você quiser o seu código funcinonando com o MySQL 4.0 e MySQL 4.1.1

```
char *xxx_reset(UDF_INIT *initid, UDF_ARGS *args,
                char *is_null, char *error);
```

Esta função é chamada quando o MySQL encontra a primiera linha em um novo grupo. Na função você deve zerar quaisquer variáveis sumárias internas e então definir o argumento dados como o primeiro argumento no grupo.

Em muitos casos isto é implementado internamente zerando todas as variáveis (por exemplo, chamando `xxx_clear()` e então chamando `xxx_add()`.

A seguinte função só é exigida pelo MySQL 4.1.1 e acima:

```
char *xxx_clear(UDF_INIT *initid, char *is_null, char *error);
```

Esta função é chamada quando o MySQL precisa de zerar o resumo dos resultados. Ele será chamado no começo de cada grupo novo mas também pode ser chamado para zerar os valores para uma consulta que não tiver registros coincidentes. `is_null` será definido para apontar para `CHAR(0)` antes de chamar `xxx_clear()`.

Você pode usar o ponteiro `error` para armazenar um byte se alguma coisa der errado.

```
char *xxx_add(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Esta função é chamada por todas as linhas que pertencem ao mesmo grupo, exceto na primeira linha. Nesta você deve adicionar o valor em UDF_ARGS a sua variavel sumária interna.

A função `xxx()` deve ser declarada da mesma forma que você define uam função UDF simples. Veja Seção 14.2.2.1 [Chamando UDF], Página 902.

A função é chamada quando todas as linhas no grupo tem sido processada. Normamente você nunca deve acessar a variável `args` aqui mas retornar o seu valor baseado em sua variável sumária interna.

Todos os argumentos processados em `xxx_reset()` e `xxx_add()` devem ser feito de forma idêntica as UDF's normais. Veja Seção 14.2.2.3 [Argumentos UDF], Página 904.

O tratamento do valor de retorno em `xxx()` deve ser feito de forma idêntica a uma UDF normal. Veja Seção 14.2.2.4 [Valores de retorno UDF], Página 906.

O argumento ponteiro para `is_null` e `error` é o mesmo para todas as chamadas `xxx_reset()`, `xxx_clear()`, `xxx_add()` e `xxx()`. Você pode utilizar isto para lembrar que você obteve um erro ou se a função `xxx()` deve retornar `NULL`. Note que você não deve armazenar uma string em `*error`! Ela é um parâmetro de apenas 1 byte!

`is_null` é zerado para cada grupo (antes de chamar `xxx_clear()`). `error` nunca é zerado.

Se `isnull` ou `error` são definidos depois de `xxx()` então o MySQL retornará `NULL` como o rsultado para a função do grupo.

14.2.2.3 Processando Argumentos

O parâmetro `args` aponta para uma estrutura `UDF_ARGS` que tem os mambros listados abaixo:

`unsigned int arg_count`
: O número de argumentos. Verifique o valor na função de inicialização se você quiser que ssua função seja chamada com um número específico de argumentos. For exemplo:

```
if (args->arg_count != 2)
{
    strcpy(message,"XXX() requires two arguments");
    return 1;
}
```

`enum Item_result *arg_type`

Os tipos para cada argumento. Os valores de tipos possíveis são `STRING_RESULT`, `INT_RESULT`, e `REAL_RESULT`.

Para ter certeza que os argumentos são de um tipo dado e retornar um erro se não forem, verifique o vetor `arg_type` na função de inicialização. Por exemplo:

```
if (args->arg_type[0] != STRING_RESULT ||
    args->arg_type[1] != INT_RESULT)
{
    strcpy(message,"XXX() requires a string and an integer");
    return 1;
}
```

Como uma alternativa para exigir que os argumentos de sua função sejam de um tipo específico, você pode usar a função de inicialização para definir o elemento `arg_type` com o tipo que você quiser. Isto faz com que o MySQL converta argumentos para aqueles tipo a cada chamada de `xxx()`. Por exemplo, para fazer conversão dos dois primeiros argumentos para string e integer, faça isto com `xxx_init()`:

```
args->arg_type[0] = STRING_RESULT;
args->arg_type[1] = INT_RESULT;
```

`char **args`

`args->args` informa a função de inicialização sobre a natureza geral dos argumentos chamados com sua função. Para um argumento constante `i`, `args->args[i]` aponta para o valor do argumento. (Veja abaixo sobre instruções de como acessar o valor de forma apropriada). Para um argumento não constante, `args->args[i]` é `0`. Um argumento constante é uma expressão é uma expressão que utiliza apenas constante, tais como `3` ou `4*7-2` ou `SIN(3.14)`. Um argumento não constante é uma expressão que refere a valores que podem alterar a cada linha, tais como nomes de coluna ou funções que são chamadas com argumentos não contantes.

Para cada chamada da função principal, `args->args` contém os argumentos atuais que são passados pela linhas sendo processadas atualmente.

As funções podem se referir a um argumento `i` como a seguir:

- Um argumento do tipo `STRING_RESULT` é dado como um apontador string mais um tamanho, para permitir o tratamento de dados binários de tamanho arbitrário. Os conteúdo da string estão disponíveis como `args->args[i]` e o tamanho da string é `args->lengths[i]`. Você não deve assumir aue as strings são terminadas em null.
- Para um argumnto do tipo `INT_RESULT`, você deve converter `args->args[i]` para um valor `long long`:

  ```
  long long int_val;
  int_val = *((long long*) args->args[i]);
  ```

- Para um argumento do tipo `REAL_RESULT`, você deve converter `args->args[i]` para um valor `double`:

  ```
  double    real_val;
  ```

```
real_val = *((double*) args->args[i]);
```

`unsigned long *lengths`

Para a função de inicialização, o vetor `lengths` indica o tamanho máximo da string para cada argumento. Você não deve alterá-los. Para cada chamada da função principal, `lengths` contém o tamanho atual de quaisquer argumentos string que são passados para a linha sendo processada atualmente. Para argumentos do tipo `INT_RESULT` ou `REAL_RESULT`, `lengths` ainda contém o tamanho máximo do argumento (como para a função de inicialização).

14.2.2.4 Valor de Retorno e Tartamento de Erros

A função de inicialização deve retornar `0` se nenhum erro ocorrer e `1` em outro caso. Se ocorrer um erro, `xxx_init()` deve armazenar uma mensagem de erro terminada em null no parâmetro `message`. A mensagem será retornada ao cliente. O buffer de mensagens tem `MYSQL_ERRMSG_SIZE` caracteres, mas você deve tentar manter a mensagem com menos que 80 caracteres assim ela cabe na tela de terminal padrão.

O valor de retorno de uma função principal `xxx()` é o valor da função, para funções `long long` e `double`. Uma função string deve retornar um ponteiro ao resultado e armazenar o tamanho da string no argumento `length`.

Definá-os ao conteúdo e tamanho do valor de retorno. Por exemplo:

```
memcpy(result, "result string", 13);
*length = 13;
```

O buffer `result` que é passado para o cálculo da função é de 255 bytes. Se o seu resultado couber nele, você não terá que se preocupar com alocação de memória para os resultados.

Se a sua função string precisar retornar uma string maior que 255 bytes, você deve alocar o espaço para ela com `malloc()` em sua função `xxx_init()` ou sua função `xxx()` e liberá-la em sua função `xxx_deinit()`. Você pode armazenar a memória alocada na posição `ptr` na estrutura `UDF_INIT` para ser reutilizado por chamadas `xxx()` futuras. Veja Seção 14.2.2.1 [Chamadno UDF], Página 902.

Para indicar um valor de retorno de `NULL` na função principal, defina `is_null` com `1`:

```
*is_null = 1;
```

Para indicar um erro retornado na função principal, atribua `1` ao parâmetro `error`:

```
*error = 1;
```

Se `xxx()` definir `*error` com `1` para qualquer linha, o valor da função é `NULL` para a linha atual e qualquer linha subsequente processada pela instrução na qual `XXX()` foi chamado. (`xxx()` nem mesmo será chamado para linhas subsequentes.) **Nota**: na versão do MySQL anterior a 3.22.10, você deve configurar `*error` e `*is_null`:

```
*error = 1;
*is_null = 1;
```

14.2.2.5 Compilando e Instalando Funções Definidas Por Usuário

Arquivos implementando UDFs devem ser compilados e instalados na máquina onde o servidor está sendo executado. Este processo é descrito abaixo pelo arquivo UDF exemplo

'udf_example.cc' que é incluído na distribuição fonte do MySQL. Este arquivo contém as seguintes funções:

- `metaphon()` retorna uma string metafonica do argumento string. Ela é algo como uma string soundex, mas é mais voltada para o inglês.
- `myfunc_double()` retorna a soma de valores ASCII de caracteres e seus argumentos, dividido pela soma de tamanho de seus argumentos.
- `myfunc_int()` retorna a soma do tamanho de seus argumentos.
- `sequence([const int])` retorna uma sequência iniciando a partir de um número dado ou 1 se nenhum número for fornecido.
- `lookup()` retorna o IP de um nome de máquina.
- `reverse_lookup()` retorna o nome de mauina para um número IP. A função pode ser chamada com uma string `"xxx.xxx.xxx.xxx"` ou quatro números.

A arquivo carregável dinamicamente deve ser compilado como um arquivo objeto compartilhável usando um comando como este:

```
shell> gcc -shared -o udf_example.so myfunc.cc
```

Você pode encontrar facilmente as opções de compilador corretas para seu sistema executando este comando no diretório 'sql' da sua árvore de fonte MySQL:

```
shell> make udf_example.o
```

Você deve executar comando de compilador similar àquele que o `make` mostra, exceto que você deve remover a opção `-c` próxima ao fim da linha e adicionar `-o udf_example.so`. (Em alguns sistemas você pode precisar deixar o comando `-c`.)

Uma vez que você tenha compilado um objeto compartilhado contendo UDFs, você deve instalá-lo e avisar o MySQL sobre ele. Compilar um objeto compartilhado de 'udf_example.cc' produz um arquivo com nome parecido com 'udf_example.so' (o nome exato pode variar de plataforma para plataforma). Copie este arquivo para algum diretório procurado com o ligador dinâmico `ld`, tal como '/usr/lib' ou adicione o diretório no qual você colocou o objeto compartilhado ao arquivo de configuração do ligador (e.g. '/etc/ld.so.conf').

Em muitos sistemas você pode as variáveis de ambiente `LD_LIBRARY` ou `LD_LIBRARY_PATH` para apontar para o diretório onde se encontra os seus arquivos de funções UDF. A página `dlopen` do manual diz a você quais variáveis você deve utilizar em seu sistema. Você deve configurar isto nos scripts de inicialização `mysql.server` ou `mysqld_safe` e reiniciar o `mysqld`.

Depois da biblioteca ser instalada, notifique `mysqld` sobre as novas funções com estes comandos:

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_double RETURNS REAL SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_int RETURNS INTEGER SONAME "udf_example.so";
mysql> CREATE FUNCTION lookup RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION reverse_lookup
    ->        RETURNS STRING SONAME "udf_example.so";
mysql> CREATE AGGREGATE FUNCTION avgcost
    ->        RETURNS REAL SONAME "udf_example.so";
```

Funções podem ser deletadas utilizando-se `DROP FUNCTION`:

```
mysql> DROP FUNCTION metaphon;
mysql> DROP FUNCTION myfunc_double;
mysql> DROP FUNCTION myfunc_int;
mysql> DROP FUNCTION lookup;
mysql> DROP FUNCTION reverse_lookup;
mysql> DROP FUNCTION avgcost;
```

As instruções `CREATE FUNCTION` e `DROP FUNCTION` atualizam a tabela de sistema `func` no banco de dados `mysql`. O nome da função, tipo e biblioteca compartilhada são salvas na tabela. Você deve ter os privilégios `INSERT` e `DELETE` para o banco de dados `mysql` para criar e deletar funções.

Você não deve usar `CREATE FUNCTION` para adicionar uma função que já tenha sido criada. Se você precisar reinstalar uma função, você deve removê-la com `DROP FUNCTION` e então reinstalá-la com `CREATE FUNCTION`. Você precisaria fazer isto, por exemplo, se você recompilar uma nova versão da sua função, assim o `mysqld` obtem a nova versão. Por outro lado, o servidor continuará a utilizar a versão antiga.

Funções ativas são recarregadas a cada vez que o servidor inicia, a menos que você inicie `mysqld` com a opção `--skip-grant-tables`. Neste caso, a a inicialização de UDF é ignorada e as UDFs ficam indisponíveis. Uma função ativa é aquela que carregada com `CREATE FUNCTION` e não removida com `DROP FUNCTION`.)

14.2.3 Adicionando uma Nova Função Nativa

O procedimento para adicionar uma nova função nativa é descrito aqui. Note que você não pode adicionar funções nativas a distribuição binária porque o procedimento envolve modificação no código fonte do MySQL. Você deve compilar o MySQL de uma distribuição fonte. Note também que se você migrar para outra versão do MySQL (por exemplo, quando uma nova versão é liberada), você precisará repetir o procedimento com a nova versão.

Para adicionar uma função MySQL nativa, siga estes passos:

1. Adicionr uma linha a '`lex.h`' que defina o nome da função no vetor `sql_functions[]`.
2. Na função protótipo é simples (utilize apenas zero, um, dois ou três argumentos), você deve especificar SYM(FUNC_ARG#) em lex.h (onde # é o número de argumentos) como o segundo argumento no vetor `sql_functions[]` e adicionar uma função que cria um objeto de função em '`item_create.cc`'. De uma olhada em `"ABS"` e `create_funcs_abs()` para um exemplo disto.

   Se o protótipo da função for complicado (por exemplo, tiver um número variável de argumentos), você deve adicionar duas linhas a '`sql_yacc.yy`'. Uma indica o símbolo pre-processador que o `yacc` deve difinir (isto deve ser adicionado no começo do arquivo). Então defina os parâmetros da função e adicione um "item" com estes parâmetros a regra `simple_expr` do analizador. Por exemplo, verifique todas as acorrências de `ATAN` em '`sql_yacc.yy`' para ver como ele é feito.
3. Em '`item_func.h`', declare uma classe herdada de `Item_num_func` ou `Item_str_func`, dependendo se sua função retorna um número ou uma string.
4. Em '`item_func.cc`', adicione uma das seguintes declarações, dependendo se você está definindo uma função numérica ou string:

```
double   Item_func_newname::val()
longlong Item_func_newname::val_int()
String  *Item_func_newname::Str(String *str)
```

Se você herdar seu objeto de qualquer um dos itens padrões (como `Item_num_func`), você provavelmente só deverá definir uma das funções acima e deixar os objetos pais cuidar das outras funções. Por exemplo, a classe `Item_str_func` define uma função `val()` que executa `atof()` no valor retornado por `::str()`.

5. Você também deve, provavelmente, definir a seguinte função objeto:

   ```
   void Item_func_newname::fix_length_and_dec()
   ```

   Esta função deve pelo menos calcular `max_length` baseado nos argumentos dados. `max_length` é o número máximo de caracteres que a função pode retornar. Esta função também deve definir `maybe_null = 0` se a função principal não puder retornar um valor `NULL`. A função pode verificar se algum dos argumentos da função pode retornar `NULL` verificando a variável de argumentos `maybe_null`. Você pode dar uma olhada em `Item_func_mod::fix_length_and_dec` para um exemplo típico de como fazer isto.

Todas as funções devem ser seguras com thread (em outras palavras, não utilize qualquer variável global ou estática nas funções sem protege-las com mutexes).

Se você retornar `NULL`, de `::val()`, `::val_int()` ou `::str()` você deve definir `null_value` com 1 e retornar 0.

Para funções objetos `::str()`, existem algumas considerações adicionais das quais você deve estar ciente:

- O arguemto `String *str` fornece um buffer string que pode ser utilizado para guardar o resultado. (Para mais informações sobre o tipo `String`, dê uma olhada no arquivo '`sql_string.h`'.)
- A função `::str()` deve retornar a string que guarda o resultado ou `(char*) 0` se o resultado é `NULL`.
- Todas as funções string atuais tentam evitar a alocação de memória a menos que seja absolutamente necessário!

14.3 Adicionado Novos Procedimentos ao MySQL

No MySQL, você pode definir um procedimento em C++ que pode acessar e modificar os dados em uma consulta antes que ela seja enviada ao cliente. A modificação pode ser feita linha a linha ou a nivel `GROUP BY`.

Nós criamos um procedimento exemplo no MySQL Versão 3.23 para mostrar o que pode ser feito.

Adicionalmente recomendamos que você de uma olhada em `mylua`. Com isto você pode utilizar a linguagem LUA para carregar um procedimento em tempo de execução no `mysqld`.

14.3.1 Análise de Procedimento

`analyse([max elements,[max memory]])`

Este procedimento é definido em '`sql/sql_analyse.cc`'. Ele examina o resultado de sua consulta e retorna uma análise do resultado:

- `max elements` (padrão 256) é o número máximo de valores distintos que `analyse` notificará por coluna. Isto é utilizado por `analyse` para verificar se o tipo ótimo da coluna deve ser do tipo `ENUM`.
- `max memory` (padrão 8192) é a memória máxima que `analyse` deve alocar por coluna enquanto tenta encontrar todos os valores distintos.

```
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
```

14.3.2 Escrevendo um Procedimento

No momento, a única documentação sobre isto é o código fonte.

Você pode encontrar todas as informações sobre procedimentos examinando os seguintes arquivos:

- '`sql/sql_analyse.cc`'
- '`sql/procedure.h`'
- '`sql/procedure.cc`'
- '`sql/sql_select.cc`'

Apêndice A Problemas e Erros Comuns

Este capítulo lista alguns problemas e mensagens de erro comuns que os usuários encontram. Você aprenderá como entender o problema e o que fazer para resolvê-lo. Você também encontrará soluções apropriadas para alguns prblemas comuns.

A.1 Como Determinar o Que Está Causando Problemas

Quando você encontrar problemas, a primeira coisa que você deve fazer é descobrir qual o programa / parte do equipamento está causando problema:

- Se você tiver um dos seguintes sintomas, então é provavel que haja um problema de hardware (como memória, placa mãe, CPU ou disco rígido) ou kernel:
  - O teclado não funciona. Isto normalmente pode ser verificado pressionando CAPS LOCK. Se a luz do CAPS LOCK não alterar, você deverá trocar o seu teclado. (Antes de fazer isto, você deve tentar reiniciar o seu computador e verificar todos os cabos do teclado.)
  - O ponteiro do mouse não move.
  - A máquina não responde ao ping de uma máquina remota.
  - Diferente, programas não relacionados não comportam corretamente.
  - Se o seu sistema reiniciar inesperadamente (um programa de nível do usuário **nunca** deve finalizar o seu sistema).

  Neste caso você deve inciar verificando todos os seus cabos e executando alguma ferramenta de diagnóstico para verificar o seu hardware. Você também deve verificar se existem patches, atualizações ou service packs para o seu sistema operacional que poderiam resolver o seu problema. Verifique também que todas as suas bibliotecas (como glibc) estão atualizadas.

  Sempre é bom usar uma máquina com memória ECC para descobrir problemas de memória antecipadamente.
- Se o seu teclado está travado, você deve estar apto a consertá-lo logando em sua máquina a partir de outra máquina e executando `kbd_mode -a` nela.
- Por favor, examine o seu arquivo de log do sistema (/var/log/messages ou similar) procurando pela razão de seus problemas. Se você acha que o problema está no MySQL então você deve examinar o arquivo de log do MySQL. Veja Seção 4.10.4 [Log binário], Página 374.
- Se você acha que você não tem problema de hardware, você deve tentar encontrar qual o programa que está causando problemas.

  Tente usar `top`, `ps`, `taskmanager`, ou algum programa parecido, para verificar qual programa está utilizando toda a CPU oui travando a máquina.
- Verifique com `top`, `df`, ou um programa similar se você excedeu a quantidade de memória, espaço em disco, arquivos abertos ou algum outro recurso crítico.
- Se o problema é algum processo em execução, você sempre pode tentar matá-lo. Se ele não quiser morrer, provavelmente há um bug em seu sistema operacional.

Se depois de você examinar todas as outras possibilidades e você tiver concluído que é o cliente MySQL ou o servidor MySQL que está causando problemas, é hora de fazer um relatório de erro para a nossa lista de emails ou nossa equipe de suporte. No relatório de erro, tente dar uma descrição bem detalhada de como o sistema se comporta e o que você acha que está acontecendo. Você também deve dizer porque você acha que é o MySQL que esta causando problemas. Lev em consideração todas as situações neste capítulo. Indique qualquer problema exatamente como ele aparece quando você examina o seu sistema. Use o método 'cortar e colar' para qualquer saída e/ou mensagem de erro do programa e/ou arquivos de log!

Tente descrever em detalhes qual programa não está funcionando e todos os sintomas que você vê! Nós recebemos muitos relatórios de erros que apenas indicavam "o sistema não funciona". Isto não nos fornece qualquer informação sobre o que poderia ser o problema.

Se um programa falhar, sempre é útil saber:

- O programa em questão realizou um operação de segmentation fault (core dumped)?
- O program aesta consumindo toda a CPU? Verifique com `top`. Deixe o programa rodar por um tempo. Ele pode estar avaliando algo pesado.
- Se é o servidor `mysqld` que está causando problemas, você pode fazer um `mysqladmin -u root ping` ou `mysqladmin -u root processlist`?
- O que o progrma cliente diz (tente com `mysql`, por exemplo) quando você tenta conectar ao servidor MySQL? O cliente travou? Você obteve qualquer saída do programa?

Quando enviar um relatório de erro, você deve seguir o que é descrito neste manual. Veja Seção 1.7.1.2 [Fazendo perguntas], Página 35.

A.2 Erros Comuns Usando o MySQL

Esta seção lista alguns erros que os usuários obtém frequqntemente. Você encontrará descrições de erros e como resolvê-los aqui.

A.2.1 Erro: `Access Denied`

Veja Seção 4.3.12 [Acesso Negado], Página 250. Veja Seção 4.3.6 [Privilégios], Página 232.

A.2.2 Erro: `MySQL server has gone away`

Esta seção também cobre o erro relacionado sobre `perda de conexão com o servidor durante uma consulta`.

A razão mais comum para o erro `MySQL server has gone away` é que o servidor esgotou o tempo limite e fechou a conexão. Por padrão, o servidor fecha uma conexão depois de 8 horas se nada aconctecer. Você pode alterar o tempo limite configurando a variável `wait_timeout` quando você iniciar o `mysqld`.

Outra razão comum para receber o erro `MySQL server has gone away` é porque você executou um "fechar" em sua conexão MySQL a então tentou executar uma consulta na conexão fechada.

Se você tiver um script, você só tem que executar a consulta novamente para o cliente reconectar autometicamente.

Você normalmente pode obter os seguintes códigos de erros neste caso (qual você obterá dependerá do SO):

Código de erro	Descrição
`CR_SERVER_GONE_ERROR`	O cliente não pode enviar um pedido ao servidor.
`CR_SERVER_LOST`	O cliente não obteve um erro ao escrever no servidor, mas não obteve uma resposta completa (ou nenhuma resposta) a seu pedido.

Você também irá obter este erro se alguém tiver matado a thread em execução com `kill #threadid#`.

Você pode verificar que o MySQL não morreu executando `mysqladmin version` e examinando o tempo em execução. Se o problema é que o mysqld falhou você deve descobrir a razão da falha. Você deve neste caso iniciar verificando se executar a consulta novamente irá finalizar o MySQL novamente. Veja Seção A.4.1 [Falhas], Página 925.

Você também pode obter estes erros se você enviar uma consulta incorreta ou muito grande ao servidor. Se `mysqld` recebe um pacote muito grande ou fora de ordem. ele assume que alguma coisa saiu errado com o cliente e fecha a conexão. Se você precisa de grandes consultas (por exemplo, se você está trabalhando com grandes colunas `BLOB`), você pode aumentar o limite da consulta iniciando o `mysqld` com a opção `-O max_allowed_packet=#` (padrão 1M). A memória extra é alocada sobre demanda, assim o `mysqld` alocará mais memória apenas quando você executar uma grande consulta ou quando o `mysqld` deve retornar um grande registro de resultado!

Você também obterá uma conexão perdida se você estiver enviando um pacote >= 16M e se seu cliente for mais antigo que a versão 4.0.8 e a versão do seu servidor é 4.0.8 e acima ou vice versa.

Se você quiser fazer um relatório de erros descreendo este prolema, esteja certo de ter incluído as seguintes informações:

- Informe se o MySQL morreu ou não. (Você pode encontrar into no arquivo `hostname.err`). Veja Seção A.4.1 [Falhas], Página 925.
- Se uma cosulta específica matar o `mysqld` e as tabelas envolvidas foram verificadas com `CHECK TABLE` antes que você fizesse a consulta, você pode fazer um caso de teste para isto? Veja Seção E.1.6 [Caso de testes reproduzíveis], Página 1084.
- Qual é o valor da variável `wait_timeout` no servidor MySQL? `mysqladmin variables` lhe dá o valor destas variáveis.
- Você tentou executar `mysqld` com `--log` e verificou se a consulta executada apareceu no log?

Veja Seção 1.7.1.2 [Fazendo perguntas], Página 35.

A.2.3 Erro: `Can't connect to [local] MySQL server`

Um cliente MySQL em Unix pode conectar ao servidor `mysqld` de dois modos diferentes: sockets Unix, que conectam através de um arquivo no sistema de arquivos (padrão '`/tmp/mysqld.sock`') ou TCP/IP, que conecta através um número de porta. Sockets Unix

são mais rápidos que TCP/IP mas só podem ser usados quando conectados ao servidor no mesmo computador. Sockets Unix são usados se você não especificar um nome de máquina ou se você especificar o nome de máquina especial `localhost`.

No Windows, se o servidor `mysqld` está rodando no 9x/Me, você só pode conectar via TCP/IP. Se o servidor estiver rodando no NT/2000/XP e o mysqld é iniciado com `--enable-named-pipe`, você também pode conectar com named pipes. O nome do named pipes é MySQL. Se você não der um nome de máquina quando conectar ao `mysqld`, um cliente MySQL tentará conectar primeiro ao named pipe, e se isto não funcionar ele irá conectar a porta TCP/IP. Você pode forçar o uso de named pipes no Windows usando `.` como nome de máquina.

O erro (2002) `Can't connect to ...` normalmente significa que não há um servidor MySQL rodando no sistema ou que você está usando um arquivo socket ou porta TCP/IP errado ao tentar conectar so servidor `mysqld`.

Inicie verificando (usando `ps` ou gerenciador de tarefas do Windows) que há um processo chamado `mysqld` executando em seu sistema! Se não houver nenhum processo `mysqld`, você deve iniciar um. Veja Seção 2.4.2 [Iniciando o servidor], Página 116.

Se um processo `mysqld` estiver em execução, você pode verificar o servidor tentando estas diferentes conexões (o número da porta e o caminho do socket devem ser diferente em sua consiguração, é claro):

```
shell> mysqladmin version
shell> mysqladmin variables
shell> mysqladmin -h `hostname` version variables
shell> mysqladmin -h `hostname` --port=3306 version
shell> mysqladmin -h 'ip for your host' version
shell> mysqladmin --protocol=socket --socket=/tmp/mysql.sock version
```

Note o uso de aspas para traz em vez de aspas para frente com o comando `hostname`; isto provoca a saída de `hostname` (que é, o nome de máquina atual) para ser substituído no comando `mysqladmin`.

Aqui estão algumas razões pela quais o erro `Can't connect to local MySQL server` pode ocorrer:

- `mysqld` não está rodando.
- Você está rodando em um sistema que usa MIT-pthreads. Se você estiver executando em um sistema que não possui threads nativas, o `mysqld` usa o pacote MIT-pthreads. Veja Seção 2.2.3 [Qual SO], Página 78. No entanto, nem todas as versões de MIT-pthreads suportam sockets Unix. Em um sistema sem suporte a sockets você sempre deve especificar o nome de máquina explicitamente ao conectar ao servidor. Tente usar este comando para verificar a conexão com o servidor:

  ```
  shell> mysqladmin -h `hostname` version
  ```

- Alguém removeu o socket Unix que o `mysqld` utiliza (por padrão '`/tmp/mysqld.sock`'). Você deve ter um trabalho `cron` que remove o socket MySQL (por exemplo, um trbalhoque remove arquivos antigos do diretório '`/tmp`'). Você sempre pode executar `mysqladmin version` e verificar que o socket que o `mysqladmin` está tentando usar realmente existe. A correção neste caso é alterar o trabalho `cron` para não remover '`mysqld.sock`' ou para colocar o socket em outro local. Veja Seção A.4.5 [Problemas com mysql.sock], Página 929.

- Você iniciou o servidor `mysqld` com a opção `--socket=/path/to/socket`. Se você alterar o caminho do socket para o servidor, você também deve notificar o cliente MySQL sobre o novo caminho. Você pode fazer isto fornecendo o caminho do socket como um argumento para o cliente. Veja Seção A.4.5 [Problemas com mysql.sock], Página 929.
- Você está usando Linux e uma thread finalizou (core dumped). Neste caso você deve matar as outras threads `mysqld` (por exemplo, com o script `mysql_zap` antes de você poder iniciar um novo servidor MySQL. Veja Seção A.4.1 [Falhas], Página 925.
- Você pode não ter privilégios de leitura e escrita tanto no diretório que guarda o arquivo de socket quanto no próprio arquivo de socket. Neste caso você deve mudar o privilégio do diretório/arquivo ou reiniciar `mysqld` para que ele use um diretorio que você possa utilizar.

Se você obter a mensagem de erro `Can't connect to MySQL server on alguma_maquina`, você pode tentar o seguinte para descobrir qual é o problema:

- Verifique se o servidor está funcionando fazendo `telnet seu-servidor num-porta-tcp-ip` e pressione Enter algumas vezes. Se houver um servidor MySQL em execução nesta porta você deve obter uma resposta que inclui o número da versão do servidor MySQL em execução. Se você obter um erro como `telnet: Unable to connect to remote host: Connection refused`, então não há nenhum servidor rodando na porta dada.
- Tente conectar ao daemon `mysqld` na máquina local e verifique a porta TCP/IP que o `mysqld` está configurado para usar (variável `port`) com `mysqladmin variables`.
- Verifique se o seu servidor `mysqld` não foi iniciado com a opção `--skip-networking`.

A.2.4 Erro: `Client does not support authentication protocol`

O MySQL 4.1 usa um protocolo de autenticação baseado em um algorítmo de hashing de senha que é incompatível com aquele usado por outros clientes. Se você atualizar o servidor para a versão 4.1, tentar se conectar a ele com um cliente mais antigo pode falhar com a seguinte mensagem:

```
shell> mysql
Client does not support authentication protocol requested
by server; consider upgrading MySQL client
```

Para resolver este problema você deve fazer um dos seguintes:

- Atualizar todos os progrmas clientes para usar a biblioteca cliente 4.1.1 ou mais nova.
- Use uma conta com uma senha antiga ao conectar em clientes anteriores ao 4.1.
- Reset o usuário que precisa de um cliente anterior ao 4.1 para usar a senha antiga:

```
mysql> UPDATE user SET Password = OLD_PASSWORD('mypass')
    -> WHERE Host = 'some_host' AND User = 'some_user';
mysql> FLUSH PRIVILEGES;
```

- Diga ao servidor para usar o algoritmo de hashing de senha antigo:
  1. Inicie o mysqld com `--old-passwords`.
  2. Defina a senha para todos os usuários que tenham senha longa. Você pode encontrar estes usuários com:

```
SELECT * FROM mysql.user WHERE LEN(password) > 16;
```

Para mais informações sobre hash de senha e autenticação, veja Seção 4.3.11 [Password hashing], Página 245.

A.2.5 Erro: `Host '...' is blocked`

Se você obter um erro como este:

```
Host 'hostname' is blocked because of many connection errors.
Unblock with 'mysqladmin flush-hosts'
```

significa que o `mysqld` obteve diversos (`max_connect_errors`) pedidos de conexão da máquina `'hostname'` e que foram interrompidos no eio. Depois de `max_connect_errors` pedidos com falhas o `mysqld` assume que algo está errado (como um attack de um cracker), e bloqueia o site para tais conexões até alguém executar o comando `mysqladmin flush-hosts`.

Por padrão, o `mysqld` bloqueia um host depois de 10 erros de conexão. Você pode facilmente ajustar isto iniciando o servidor assim:

```
shell> mysqld_safe -O max_connect_errors=10000 &
```

Note que se você obter esta mensagem de erro para uma dada máquina, você deve primeiramente verificar se não há nada errado com a conexão TCP/IP desta máquina. Se sua conexão TCP/IP não estiver funcionando, não será nada bom aumentar o valor da variável `max_connect_errors`!

A.2.6 Erro: `Too many connections`

Se você obter o erro `Too many connections` quando vacê tentar se conectar ao MySQL, isto significa que já existe `max_connections` clientes conectados ao servidor `mysqld`.

Se você precisar de mais conexões do que o padrão (100), então você deve reiniciar o `mysqld` com um valor maior para a variável `max_connections`.

Note que atualmente o `mysqld` permite que (`max_connections`+1) clientes se conectem. A última conexão é reservada para um usuário com o privilégio `SUPER`. Ao não dar este privilégio a usuários normais (eles não precisam dele), um administrador com este privilégio pode logar e utilizar `SHOW PROCESSLIST` para descobrir o que pode estar errado. Veja Seção 4.6.8.6 [SHOW PROCESSLIST], Página 320.

O número máximo de conexões MySQL depende de quão boa é a biblioteca de threads na dada plataforma. Linux ou Solaris devem estar aptos a suportar 500-1000 conexões simultâneas, dependendo de quanta RAM você tem e do que o cliente está fazendo.

A.2.7 Erro: `Some non-transactional changed tables couldn't be rolled back`

Se você obter o erro/aviso: `Warning: Some non-transactional changed tables couldn't be rolled back` ao tentar fazer um `ROLLBACK`, isto significa que algumas das tabelas que você utiliza na transação não suportam transações. Estas tabelas não transacionaisn não serão afetadas pela instrução `ROLLBACK`.

O caso mais comum em que isto acontece é quando você tenta criar uma tabela de um tipo que não é suportado por seu binário `mysqld`. Se o `mysqld` não suporta um tipo de tabela (ou se o tipo de tabela está disabilitado por uma opção de inicialização), ele criará a tabela com o tipo mais comumente usado em suas outras tabelas, que é provavelmente o `MyISAM`.

Você pode verificar o tipo de uma tabela fazendo:

`SHOW TABLE STATUS LIKE 'nome_tabela'`. Veja Seção 4.6.8.2 [SHOW TABLE STATUS], Página 304.

Você pode verificar as extensão que seu binário `mysqld` suporta com:

`show variables like 'have_%'`. Veja Seção 4.6.8.4 [SHOW VARIABLES], Página 309.

A.2.8 Erro: `Out of memory`

Se você executar uma consulta e obter algo como o seguinte erro:

```
mysql: Out of memory at line 42, 'malloc.c'
mysql: needed 8136 byte (8k), memory in use: 12481367 bytes (12189k)
ERROR 2008: MySQL client ran out of memory
```

note que o erro se refere ao cliente MySQL `mysql`. A razão para este erro é simplesmente que o cliente não possui memória suficente para armazenar todo o resultado.

Para solucionar o problema, primeiro verifique que sua consulta está correta. É razoável que você deva retornar tantos registros? Se for, você pode utilizar `mysql --quick`, que usa `mysql_use_result()` para retornar o resultado. Isto coloca menos carga no cliente (mas mais carga nop servidor).

A.2.9 Erro: `Packet too large`

Quando um cliente MySQL ou o servidor `mysqld` recebe um pacote maior que `max_allowed_packet` bytes, ele envia o erro `Packet too large` e fecha a conexão.

No MySQL 3.23 o maior pacote possível é 16M (devido a limites do protocolo cliente/servidor). No MySQL 4.0.1 e acima el só é limitado pela quantidade de memória que você tem no seu servidor (até um máximo teórico de 2GB).

Um pacote de conexão é uma única instrução SQL enviada ao servidor MySQL ou um única linha enviada para o cliente.

Quando um cliente MySQL ou o servidor `mysqld` obtem um pacote maior que `max_allowed_packet` bytes, ele envia o erro `Packet too large` e fecha a conexão. Com alguns clientes você também pode obter o erro `Lost connection to MySQL server during query` se o pacote de comunicação for muito grande.

Note que tanto o cliente quanto o servidor tem a sua própria variável `max_allowed_packet`. Se você quiser tratar os pacotes grandes, você tem que aumentar esta variável tanto no cliente quanto no servidor.

É seguro aumentar esta variável já que a memória só é alocada quando necessário; esta variável é mais uma precaução para pegar pacotes errados entre o cliente/servidor e também para assegurar que você use pacotes grandes acidentalemente e assim fique sem memória.

Se você estiver usando o cliente `mysql`, você pode especificar um buffer maior iniciando o cliente com `mysql --set-variable=max_allowed_packet=8M`. Outros clientes tem

métodos diferentes de configurar esta variável. Por favor, note que `--set-variable` está obsoleta desde o MySQL 4.0, em seu lugar utilize `--max-allowed-packet=8M`.

Você pode utilizar o arquivo de opção para definir `max_allowed_packet` com um tamanho maior no `mysqld`. Por exemplo, se você está esperando armazenar o tamanho total de um `MEDIUMBLOB` em uma tabela, você precisará iniciar o servidor com a opção `set-variable=max_allowed_packet=16M`.

Você também pode obter problemas estranhos com pacotes grandes se você estiver usando blobs grandes, mas você não deu para `mysqld` accesso a memória suficiente para tratar a consulta. Se você suspeita que este é o caso, tente adicionar `ulimit -d 256000` no inicio do script `mysqld_safe` e reinicie o `mysqld`.

A.2.10 Erros de Comunicação / Comunicação Abortada

A partir do `MySQL 3.23.40` você só recebe o erro de `Conexão abortada` se você iniciar o `mysqld` com `--warnings`.

Se você encontar erros como o seguinte em seu log de erro.

```
010301 14:38:23  Aborted connection 854 to db: 'users' user: 'josh'
```

Veja Seção 4.10.1 [Error log], Página 372.

Isto significa que algum dos seguintes problemas ocorreu:

- O programa cliente não chamou `mysql_close()` antes de sair.
- O cliente tem esperado mais que `wait_timeout` ou `interactive_timeout` sem fazer nenhuma requisição. Veja Seção 4.6.8.4 [`wait_timeout`], Página 309. Veja Seção 4.6.8.4 [`interactive_timeout`], Página 309.
- O programa cliente finalizou abuptamente no meio de uma transferência.

Quando o descrito acima ocorrer, a variável `Aborted_clients` do servidor é incrementeda.

A variável `Aborted_connects` do servidor é incrementeda quando:

- Quando um pacote de conexão não contém a informação correta.
- Quando o usuário não tiver privilégios para conectar ao banco de dados.
- Quando o usuário usar uma senha errada.
- Quando levar mais de `connect_timeout` segundos para obter um pacote de conexão. Veja Seção 4.6.8.4 [`connect_timeout`], Página 309.

Note que o descrito acima podia indicar que alguém está tentando derrubar o seu banco de dados.

Outras razões para problemas com Clientes abortados / Conexões abortadas.

- O uso de protocolo Ethernet com Linux, tanto half quanto full duplex. Muitos drivers de Ethernet do Linux possui este bug. Você deve testá-lo transferindo um arquivo enorme via ftp entre estas duas máquinas. Se uma transferência entra no modo de estouro-pausa-esoturo-pausa... então você está experimentando uma síndorme de duplex no Linux A única solução é trocar o modo duplex, tanto da placa de rede quanto do Hub/Switch entre full duplex e half duplex e testar os resultados para decidir qual é a melhor configuração.
- Alguns problemas com a biblioteca de threads interrompe uma leitura.

- TCP/IP mal configurado.
- Defeitos na rede, hub, switch, cabos, ... Isto pode ser diagnosticado de forma apropriada aomente através de reposição de hardware.
- `max_allowed_packet` é muito pequeno ou a consulta exige memória livre que você alocou para `mysqld`. Veja Seção A.2.9 [Packet too large], Página 917.

A.2.11 Erro: `The table is full`

exitem alguns casos diferentes nos quais você pode obter este erro:

- Você está usando um versão mais antiga do MySQL (antes da 3.23.0) quando uma tabela temporária em memória se torna maior que `tmp_table_size` bytes. Para evitar este problema, você pode utilizar a opção `-O tmp_table_size=#` para fazer o `mysqld` aumentar o tamanho da tabela temporária ou usar a opção SQL `SQL_BIG_TABLES` antes de disparar a consulta problematica. Veja Seção 5.5.6 [`SET`], Página 460.

  Você também pode iniciar `mysqld` com a opção `--big-tables`. Isto é extamente o mesmo que usar `SQL_BIG_TABLES` para toadas as consultas.

  No MySQL Versão 3.23, se uma tabelas temporárias em memória se torna maior que `tmp_table_size`, o servido automaticamente a converte para tabelas em disco `MyISAM`.
- Você está usando tabelas `InnoDB` e fica sem espaço no tablespace do `InnoDB`. Neste cado a solução é extender o tablespace do `InnoDB`.
- Você está usando tabelas `ISAM` ou `MyISAM` em um SO que só suporta arquivos de 2G e você alcançou este limite para os arquivos de dado ou índice.
- Você está usando tabelas `MyISAM` e o dado necessário ou tamanho do índice é maior que alqueles para os quais o MySQL alocou ponteiros. (Se você não especificar `MAX_ROWS` para `CREATE TABLE` o MySQL só alocará ponteriros para guardar 4G de dados).

  Você pode verificar o tamanho máximo do dados/índice fazendo

  ```
  SHOW TABLE STATUS FROM database LIKE 'nome_tabela';
  ```

  ou usando `myisamchk -dv database/nome_tabela`.

  Se este é o problema, você pode corrigí-lo fazendo algo como:

  ```
  ALTER TABLE nome_tabela MAX_ROWS=1000000000 AVG_ROW_LENGTH=nnn;
  ```

  Você só precisa especificar `AVG_ROW_LENGTH` para tabelas com campos `BLOB/TEXT` já que neste caso o MySQL não pode otimizar o espaço necessário baseado apenas no número de linhas.

A.2.12 Erro: `Can't create/write to file`

Se você obter um erro do tipo abaixo em suas consultas:

```
Can't create/write to file '\\sqla3fe_0.ism'.
```

significa que o MySQL não pode criar um arquivo temporário para o resultado no diretório temporário dado. (O erro acima é um mensagem de erro típica no Windows; a mensagem de erro do Unix é parecida.) A correção é iniciar o `mysqld` com `--tmpdir=path` ou adicionar ao seu arquivo de opção:

```
[mysqld]
tmpdir=C:/temp
```

assumindo que o diretórioe '`c:\\temp`' existe. Veja Seção 4.1.2 [Arquivos de opção], Página 216.

Verifique também o código de erro que você obteve com `perror`. Outra razão pode ser um erro de disco cheio;

```
shell> perror 28
Error code  28:  No space left on device
```

A.2.13 Erro no Cliente: `Commands out of sync`

Se você obter `Commands out of sync; you can't run this command now` no código de seu cliente, você está chamando funções do cliente na ordem errada.

Isto pode acontecer, por exemplo, se você está utilizando `mysql_use_result()` e tenta executar uma nova consulta antes de chamar `mysql_free_result()`. Isto também pode acontecer se você tentar executar duas consultas que retornam dados sem um `mysql_use_result()` ou `mysql_store_result()` entre elas.

A.2.14 Erro: `Ignoring user`

Se você obter o seguinte erro:

`Found wrong password for user: 'some_user@some_host'; ignoring user`

significa que quando o `mysqld` foi iniciado ou quando recarregiou a tabela de permissões, ele encontrou uma entrada na tabela `user` com uma senha inválida. Como resultado, a entrada é simplesmente ignorada pelo sistema de permissões.

As possíveis causas e correções para este problema:

- Você pode executar uma nova versão do `mysqld` com uma tabela `user` antiga. Você pode verificar isto executando `mysqlshow mysql user` para ver se o campo da senha é menor que 16 caracteres. Se for, você pode corrigir esta condição executando o script `scripts/add_long_password`.
- O usuário tem um senha antiga (8 caracteres) e você não iniciou o `mysqld` com a opção `--old-protocol`. Atualize o usuário na tabela `user` com uma nova senha ou reinicie o `mysqld` com `--old-protocol`.
- Você especificou uma senha na tabela de usuário `user` sem sar a função `PASSWORD()`. Use `mysql` para atualizar o usuário na tabela `user` com uma nova senha. Utilize a função `PASSWORD()`:

```
mysql> UPDATE user SET password=PASSWORD('your password')
    ->             WHERE user='XXX';
```

A.2.15 Erro: `Table 'xxx' doesn't exist`

Se você obter o erro `Table 'xxx' doesn't exist` ou `Can't find file: 'xxx' (errno: 2)`, significa que não existem tabelas no banco de dados atual com o nome `xxx`.

Note que como o MySQL utiliza diretórios e arquivos para armazenar banco de dados e tabelas, o nome de banco de dados e tabelas são **caso-sensitive**! (No Windows o nome de banco de dados e tabelas não são caso-sensitivo mas todas as referências a uma dada tabela dentro de uma consulta devem utilizar o mesmo caso!)

Você pode verificar quais tabelas existem no banco de dados atual com `SHOW TABLES`. Veja Seção 4.6.8 [SHOW], Página 302.

A.2.16 Erro: `Can't initialize character set xxx`

Se você obtr um erro do tipo:

```
MySQL Connection Failed: Can't initialize character set xxx
```

significa que é um dos seguintes problemas:

- O conjunto de caracter é multi-byte e você não tem suporte para o conjunto de caracteres no cliente.

  Neste caso você precisa recompilar o cliente com `--with-charset=xxx` ou com `--with-extra-charsets=xxx`. Veja Seção 2.3.3 [configure options], Página 98.

  Todos os bionários MySQL padrões são compilados com `--with-extra-character-sets=complex` que habilita o suporte para todos os conjuntos de caracteres multi-byte. Veja Seção 4.7.1 [Conjunto de caracteres], Página 325.
- O conjunto de caracteres é simples e não foi compilado no `mysqld` e os arquivos de definição do conjunto de caracteres não estão localizados onde o cliente esperava encontrá-los.

  Neste caso você precisa:

  - Recompilar o cliente com suporte ao conjunto de caracteres. Veja Seção 2.3.3 [configure options], Página 98.
  - Especificar para o cliente onde o arquivo de definição do conjuntos de caracteres está. Para muitos clientes você pode fazê-lo com a opção `--character-sets-dir=path-to-charset-dir`.
  - Copie o arquivo de definição de caracteres no caminho onde o cliente espera que eles estejam.

A.2.17 Arquivo Não Encontrado

Se você obter `ERROR '...' not found (errno: 23)`, `Can't open file: ... (errno: 24)`, ou qualquer outro erro com `errno` 23 ou `errno` 24 no MySQL, significa que você não alocou descritores de arquivo suficiente para o MySQL. Você pode usar o utilitário `perror` para obter uma descrição sobre o que o número de erro significa:

```
shell> perror 23
File table overflow
shell> perror 24
Too many open files
shell> perror 11
Resource temporarily unavailable
```

O problema aqui é que `mysqld` está tentando manter aberto muitos arquivos simultanemanete. Você pode também dizer para o `mysqld` não abrir muitos arquivos de uma vez ou aumentar o número de descritores de arquivos disponíveis para o `mysqld`.

Para dizer para o `mysqld` manter aberto poucos arquivos por vez, você pode tornar a cache de tabela menor usando a opção `-O table_cache=32` para `mysqld_safe` (o valor padrão é 64). Reduzindo o valor de `max_connections` também reduzirá o número de arquivos abertos (o valor padrão é 90).

Para alterar o número de descritores de arquivos disponíveis para `mysqld`, você pode usar a opção `--open-files-limit=#` para `mysqld_safe` ou `-O open-files-limit=#` para `mysqld`. Veja Seção 4.6.8.4 [`open_files_limit`], Página 309. O modo mais fácil de fazer isto é adicioar a opção ao seu arquivo de opção. Veja Seção 4.1.2 [Arquivos abertos], Página 216. Se você tiver um versão antiga do `mysqld` que não suporte isto, você pode editar o script `mysqld_safe`. Existe uma linha `ulimit -n 256` comentada no script. Você pode remover o caracter `'#'` para "descomentar" esta linha, e altere o número 256 para afetar o número de descritores de arquivos disponíveis para `mysqld`.

`ulimit` (e `open-files-limit`) podem aumentar o número de descritorese de arquivo, mas apenas até o limite imposto pelo sistema operacional. Também há um limite 'maior' que só pode ser sobrescrito se você iniciar o `mysqld_safe` ou `mysqld` como root (apenas se lembre que você também precisa usar a opção `--user=...` neste caso). Se você precisa aumentar o limite do SO no número dos descritores de arquivo disponíveis para cada processo, cosulte a documentacão para ser sistema operacional.

Note que se você rodar o shell `tcsh`, `ulimit` não funcioará! `tcsh` também relatará o valor incorreto quando você pergunta pelo limite atual! Neste caso você deve iniciar `mysqld_safe` com `sh`!

A.3 Assuntos Relacionados a Instalação

A.3.1 Problemas de Ligação com a Biblioteca do Cliente MySQL

Se você estiver ligando o seu programa e obter o erro de símbolos sem referência que iniciam com `mysql_`, como os seguintes:

```
/tmp/ccFKsdPa.o: In function ‘main’:
/tmp/ccFKsdPa.o(.text+0xb): undefined reference to ‘mysql_init’
/tmp/ccFKsdPa.o(.text+0x31): undefined reference to ‘mysql_real_connect’
/tmp/ccFKsdPa.o(.text+0x57): undefined reference to ‘mysql_real_connect’
/tmp/ccFKsdPa.o(.text+0x69): undefined reference to ‘mysql_error’
/tmp/ccFKsdPa.o(.text+0x9a): undefined reference to ‘mysql_close’
```

você deve estar apto a resolvê-los adicionando `-Lpath-to-the-mysql-library -lmysqlclient` **no final** da sua linha de ligação.

Se você obter erros de `undefined reference (referência indefinida)` para as funções `descompactadas` ou `compactadas`, adicione `-lz` **no final** sa sua linha de ligação e tente novamente!

Se você obter erros de `undefined reference (referência indefinida)` para funções que devem existir em seu sistema, como `connect`, verifique a página do man sobre a função em questão para saber quais bibiotecas você deve adicionar a sua linha de ligação!

Se você obter erros de `undefined reference (referência indefinida)` para funções que não existem em seu sistema, como o seguinte

```
mf_format.o(.text+0x201): undefined reference to `__lxstat'
```

normalmente significa que sua biblioteca é compilada em um sistema que não é 100% compatível com o seu. Neste caso você de fazer o download da última distribuição fonte do MySQL e compilá-la você mesmo. Veja Seção 2.3 [Instalação da distribuição fonte], Página 94.

Se você estiver tentando executar um programa e então obter erros de símbolos sem referência que começam com `mysql_` ou que a biblioteca do `mysqlclient` não pode encontrar, significa que seu sistema não pode encontrar a biblioteca compartilhada '`libmysqlclient.so`'.

A correção deste problema é dizer ao seu sistema para buscar onde a biblioteca esta lacolizada usando um dos seguintes métodos:

- Adicione o caminho ao diretório onde está o '`libmysqlclient.so`' à variável de ambiente `LD_LIBRARY_PATH`.
- Adicione o caminho ao diretório onde está o '`libmysqlclient.so`' à variável de ambiente `LD_LIBRARY`.
- Copie '`libmysqlclient.so`' a algum local que é pesquisado pelo seu sistema, como '`/lib`', e atualize a informação da biblioteca compartilhada executando `ldconfig`.

OUtro modo de resolver este problema é ligar o seu programa estaticamente, com `-static`, ou removendo as bibliotecas dinâmicas do MySQL antes de ligar o seu código. Na próxima vez você deve estar certo que nenhum outro programa esta usando bibliotecas dinâmicas!

A.3.2 Como Executar o MySQL Como Um Usuário Normal

O servidor `mysqld` pode ser iniciado por qualquer usuário. Para fazer com que o `mysqld` execute como um usuário `nome_usuário` do Unix, você deve fazer o seguinte:

1. Pare o servidor se ele estiver em execução (use `mysqladmin shutdown`).
2. Altere o diretório de banco de dados e arquivos para que `nome_usuário` tenha privilégios de leitura e escrita do arquivo (você pode precisar estar como o usuário `root` do Unix):

   ```
   shell> chown -R nome_usuario /caminho/para/dir_dados/mysql
   ```

   Se o diretório ou arquivos dentro do diretório de dados do MySQL são links simbolicos, você também precisará seguir estes links e alterar os diretórios e arquivos para os quais ele aponta. `chown -R` pode não seguir o link simbólico para você.
3. Inicie o servidor como o usuário `nome_usuário`, ou, se você está usando o MySQL Versão 3.22 ou mais antiga, inicie o `mysqld` como o usuário `root` do Unix e use a opção `--user=nome_usuario`. `mysqld` trocará para executar como o usuário `nome_usuário` do Unix antes de aceitar qualquer conexão.
4. Para iniciar o servidor automaticamente com o nome de usuário dado na inicialização do sistema, adicione um linha `user` que especifica o nome do usuário ao grupo `[mysqld]`

do arquivo de opções '`/etc/my.cnf`' ou o arquivo de opções '`my.cnf`' no diretório de dados do servidor. Por exemplo:

```
[mysqld]
user=nome_usuario
```

Neste ponto, seu processo `mysqld` deve estar executando bem e redondo como usuário `nome_usuario` do Unix. No entanto algo não altera: o conteúdo da tabela de permissões. Por padrão (logo depois de executar o script de instalação das tabelas de permissões `mysql_install_db`), o usuário MySQL `root` é o único com permissão para acessar o banco de dados `mysql` ou para criar ou apagar banco de dados. A menos que você tenha alterado estas permissões, elas ainda valem. Isto não deve impedí-lo de de acessar o MySQL como usuário `root` do MySQL quando você está logado como outro usuário Unix deiferente de `root`; apenas especifique a opção `-u root` ao programa cliente.

Note que acessar o MySQL como `root`, fornecendo `-u root` na linha de comando é **diferente** de de executar o MySQL como o usuário `root` do Unix,or como outro Usuário Unix. A permissão de acesso e nome de usuários do MySQL estão completamente separados dos nomes de usuário do Unix. A única conexão com os nomes de usuário do Unix é que se você não utilizar a opção `-u` quando chamr o seu programa cliene, o cliente tentará conectar usando seu nome de login do Unix como o seu nome de usuário do MySQL

Se a sua conta Unix não esta segura, você deve pelo menos colocar uma senha no usuário `root` do MySQL na tabela de acesso. Senão qualquer usuário com uma conta nesta máquina poderá executar `mysql -u root nome_bd` e fazer o que quiser.

A.3.3 Problemas com Permissões de Arquivos

Se você tiver problemas com permissões de arquivo, por exemplo, se o `mysql` enviar a seguinte mensagem de erro quando você criar uma tabela:

```
ERROR: Can't find file: 'path/with/filename.frm' (Errcode: 13)
```

então a variável de ambiente `UMASK` pode estar configurada incorretamente quando o `mysqld` inicia. O valor umask padrão é `0660`. Você pode alterar este comportamento iniciando o `mysqld_safe` como a seguir:

```
shell> UMASK=384  # = 600 em octal
shell> export UMASK
shell> /path/to/mysqld_safe &
```

Por padrão o MySQL criará o banco de dados e diretórios `RAID` com permissão tipo 0700. Você pode modificar este comportamento configurando a variável `UMASK_DIR`. Se você definir isto, novos diretórios são criados com a combinação de `UMASK` e `UMASK_DIR`. Por exemplo, se você quiser ao grupo a todos os novos diretórios, você pode fazer:

```
shell> UMASK_DIR=504  # = 770 em octal
shell> export UMASK_DIR
shell> /path/to/mysqld_safe &
```

No MySQL Versão 3.23.25 e acima, o MySQL assume que o valor para `UMASK` e `UMASK_DIR` está em octal se ele iniciar com um zero.

Veja Apêndice F [Variáveis de ambiente], Página 1092.

A.4 Assuntos Relacionados a Administração

A.4.1 O Que Fazer Se o MySQL Continua Falhando

Todas as versões do MySQL são testadas em muitas plataformas antes de serem distribuídas. Isto não significa que não existe nenhum erro no MySQL, mas significa que se houver erros, eles são poucos e podem ser difíceis de encontrar. Se você tiver u problema, sempre ajudará se você tentar encontrar exatamente o que falhou em seu sistema e assim você terá uma chance muito maior de tê-lo corrigido rapidamente.

Primeiro, você deve tentar descobrir o problema é que o daemon do `mysqld` morre ou se o seu problema é relativo ao seu cliente. Você pode verificar o quanto tempo o seu servidor `mysqld` está em execução utilizando o `mysqladmin version`. Se o `mysqld` morrer, você pode encontrar a causa disto no arquivo '`mysql-data-directory/‘nome_maquina‘.err`'. Veja Seção 4.10.1 [Registro de Erros], Página 372.

Em alguns sistemas você pode encontrar neste arquivo um stack trace de onde o `mysqld` finalizou e assim você pode resolver com `resolve_back_stack`. Veja Seção E.1.4 [Usando o stack trace], Página 1082. Note qte os valores da variável escrita no arquivo `.err` não podem sempre estar 100% corretas.

Muitas falhas do MySQL são causadas por arquivos de índices/dados corrompidos. O MySQL atualizará os dados em disco, com a chamada de sistema `write()`, depois de todas as intruções SQL e antes do ser notificado sobre o resultado. (Isto não é verdade se você estiver executando com `delay_key_write`, caso no qual apenas o dado é escrito.) Insto significa que o dado é salvo mesmo se o `mysqld` falhar, já que o SO se certificará de que o dado não descarregado esta escrito em disco. Você pode forçar o MySQL a sincronizar tudo para o disco depois de todo comando SQL inicando o `mysqld` com `--flush`.

O exposto acimo significa que normalmente você não deve ter tabelas corrompidas a menos que:

- Alguém/algo finalize o `mysqld` ou a máquina no meio de uma atualização.
- Você encontrou um bug no `mysqld` que faça com que ele finalize no meio de uma atualização.
- Alguém está manipulando os arquivos de dados/índices de fora do **mysqld** sem o bloqueio de tabela apropriado.
- Se você estiver executando muitos servidores `mysqld` no mesmo dado em um sistema que não suporta bons bloqueios de sistema de arquivos (normalmente tartando o daemon `lockd`) ou se você está executando multiplos servidores com `--skip-external-locking`
- Você tem um arquivo de dados/índices que contem muitos dados errados que deixam o `mysqld` confuso.
- Você encontrou um bug no código de armazenamento do dado. Isto não é desejável mas é possível. Neste caso você ode tentar alterar o tipo de arquivo para outro mecanismo de armazenamento usando `ALTER TABLE` em uma cópia corrigida da tabela!

Por ser muito difícil saber o motivo das falhas, tente primeiro verificar se o que está funcionando para outros está falhando com você. Por favor, tente o seguinte:

Finalize o daemon `mysqld` com `mysqladmin shutdown`, execute `myisamchk --silent --force */*.MYI` em todas as tabelas e reinicie o daemon `mysqld`. Isto irá assegurar que você está executando de um estado "limpo". Veja Cap"ptexi tulo 4 [Administrador de Banco de Dados MySQL], Página 207.

- Use `mysqld --log` e tente determinar a partir da informação no log se alguma consulta específica finalizou o servidor. Aproximadamente 95% de todos os erros são relacionados com um consulta em particular! Normalmente ela é uma das últimas consultas no arquivo de log antes do MySQL reiniciar Veja Seção 4.10.2 [Registro de consultas], Página 373. Se você puder finalizar o MySQL repetidamente com uma das consultas, mesmo quando você tiver verificado todas as tabelas logo antes de realizá-la, então você estará apto a localizar o bug e deve fazer um relatório de bug para isto! Veja Seção 1.7.1.3 [Relatório de bugs], Página 36.
- Tente fazer um caso de teste que possamos utilizar para reproduzir o problema. Veja Seção E.1.6 [Caso de testes reproduzíveis], Página 1084.
- Tente executar o teste incluso mysql-test e o benchmark do MySQL. Veja Seção 14.1.2 [Pacote de testes do MySQL], Página 896. Eles devem testar o MySQL bem. Você também pode adicionar ao benchmark um código que simule a sua aplicação! O benchmark pode ser encontrado no diretório '`bench`' na distribuição fonte ou, em uma distribuição binária, no diretório '`sql-bench`' sob o diretório de instalação do seu MySQL.
- Experimente `fork_test.pl` e `fork2_test.pl`.
- Se você configurar o MySQL para depuração, será muito mais fácil para obter informações sobre possíveis erros se alguma coisa der errado. Reconfigure o MySQL com a opção `--with-debug` ou `--with-debug=full` no `configure` e então recompile-o. Veja Seção E.1 [Depurando o servidor], Página 1079.
- Configurar o MySQL para depuração faz com que um alocador de memória seja incluído para que se possa encontrar alguns erros. Ele também fornece muita informação sobre o que está acontecendo.
- Você aplicou todas as últimas correções para o seu sistema operacional?
- Use a opção `--skip-external-locking` com o `mysqld`. Em alguns sistemas, o gerenciador de bloqueios `lockd` não funciona de forma apropriada; a opção `--skip-external-locking` faz com que `mysqld` não utilize bloqueio externo. (Isto significa que você não pode executar 2 servidores `mysqld` sobre o memo dado e que você deve ser cuidadoso ao utilizar `myisamchk`, mas pode ser instrutivo tentar a opção como teste).
- Você tentou `mysqladmin -u root processlist` quando o `mysqld` parecia estar rodando mas não respondia? Algumas vezes o `mysqld` não está <<comatose>> mesmo quando você acha que não. O problema pode ser que todas as conexões estão em uso, o pode haver algum problema interno de bloqueio. `mysqladmin processlist` normalmente estará apto a fazer uma conexão mesmo nestes casos e pode fornecer informação útil sobre o número conexões atuais e os seus estados.
- Execute o comando `mysqladmin -i 5 status` ou `mysqladmin -i 5 -r status` ou em uma janela separada para produzir estatísticas enquanto você executa outras consultas.
- Experimente o seguinte:
  1. Inicie o `mysqld` a partir do `gdb` (ou em outro depurador). Veja Seção E.1.3 [Usando gdb no mysql], Página 1081.

2. Execute o seu script de testes.
3. Imprima o <<backtrace>> e as varáveis locais nos 3 níveis mais baixos. No gdb você pode fazê-lo com o seguinte comando quando o `mysqld` falhar dentro do gdb:

```
backtrace
info local
up
info local
up
info local
```

   Com gdb você também pode examinar quais threads existem com `info threads` e troca para uma thread específica com `thread #`, onde `#` é a ID da thread.

- Tente simular a sua aplicação com um script Perl para forçar o MySQL a falhar o mudar o seu comportamento.
- Envie um relatório de bug normal. Veja Seção 1.7.1.3 [Relatório de erros], Página 36. Seja mais detalhista que o normal. Como o MySQL funciona para muitas pessoas, pode ser que as falhas resultem de algo que exista apenas em seu computador (por exemplo, um erro que é relacionado a suas bibliotecas de sistemas em particular).
- Se você tiver um problema em tabelas com registros do tamanho dinâmico e você não está usando colunas `BLOB/TEXT` (mas apenas colunas `VARCHAR`, você pode tentar alterar todas as colunas `VARCHAR` para `CHAR` com `ALTER TABLE`. Isto forçara o MySQL a usar linhas de tamanho fixo. Linhas de tamanho fixo utilizam um pouco mais de espaço extra, mas são muito mais tolerante a corrompimento.

  O código de registro dinâmico atual foi usado pela MySQL AB por pelo menos 3 anos em qualquer problema, mas por natureza os registro de tamanho dinâmico são mais propensos a erros, assim pode ser uma boa idéia tentar o exposto acima para ver se ajuda.

A.4.2 Como Recuperar uma Senha de Root Esquecida

Se você nunca definiu um senha de `root` para o MySQL, então o servidor não irá exigir uma senha para a conexão como `root`. É recomendado que sempre seja definida uma senha para cada usuário. Veja Seção 4.3.2 [Segurança], Página 229.

Se você tiver definido um senha de `root`, mas a esqueceu, você pode definir uma nova senha com o seguinte procedimento:

1. Finalize o daemon `mysqld` enviando um `kill` (não `kill -9`) para o servidor `mysqld`. O pid é armazenado em um arquivo '`.pid`', que normalmente está no diretório de banco de dados do MySQL:

```
shell> kill `cat /mysql-data-directory/hostname.pid`
```

   Você deve ser o usuário `root` do Unix ou o mesmo usuário com o qual o `mysqld` está executando para fazer isto.

2. Reinicie o `mysqld` com a opção `--skip-grant-tables`.
3. Defina uma nova senha com o comando `mysqladmin password`:

```
shell> mysqladmin -u root password 'mynewpassword'
```

4. Agora você também pode parar o `mysqld` e reiniciá-lo normalmente, ou apenas carregue a tabela de privilégios com:

```
shell> mysqladmin -h hostname flush-privileges
```

5. Depois disto, você deve estar apto para conectar usando a nova senha.

De forma alternativa, você pode definir a nova senha usando o cliente `mysql`:

1. Finalize e reinicie o `mysqld` com a opção `--skip-grant-tables` com descrito acima.
2. Conecte ao servidor `mysqld` com:

```
shell> mysql -u root mysql
```

3. Dispare os seguintes comandos no cliente `mysql`:

```
mysql> UPDATE user SET Password=PASSWORD('minhanovasenha')
    ->             WHERE User='root';
mysql> FLUSH PRIVILEGES;
```

4. Depois disto, você deve estar apto a conectar usando a nova senha.
5. Você agora pode parar o `mysqld` e reiniciá-lo normalmente.

A.4.3 Como o MySQL Trata de Discos Sem Espaço

Quando o ocorre uma condição de disco sem espaço, o MySQL faz seguinte:

- Ele verifica a cada minuto para ver se existe espaço suficiente para escrever a linha atual. Se houver espaço suficiente, ele continua como se nada tivesse aconteciso.
- A cada 6 minutos ele grava uma entrada no log de arquivo avisando sobre a condição de disco cheio.

Para aliviar o problema, você pode realizar as seguintes ações:

- Para continuar, você só tem que liberar espaço suficiente em disco para inserir todos os registros.
- Para abortar a thread, você deve enviar um `mysqladmin kill` para a thread. A thread será abortada a próxima vez que ele verificar o disco (em 1 minuto).
- Note que outra thread pode estar esperando pelas tabelas que provocaram a condição de disco cheio. Se você tiver diversas theads "bloqueadas", matar a que está esperando pela condição de disco cheio irá permitir as outras threads de continuar.

A exceção ao comportamento acima é quando você usa `REPAIR` ou `OPTIMIZE` ou quando os índices são criados em um grupo antes de um `LOAD DATA INFILE` ou depois de uma instrução `ALTER TABLE`.

Todos os comandos acima podem usar arquivos temporários grandes que por si próprios poderiam causar grandes problemas para o resto do sistema. Se o MySQL ficar sem espaço em disco enquanto faz qualquer uma das operações acima, ele removerá o arquivo temporário grande e indicara que houve falha na tabela (exceto para `ALTER TABLE`, no qual a tabela antiga ficará inalterada).

A.4.4 Onde o MySQL Armazena Arquivos Temporários

O MySQL usa o valor da variável de ambiente `TMPDIR` como caminho para o diretória que aramzena os arquivos temporários. Se você não tiver definido `TMPDIR`, o MySQL usa o padrão do sistema, que normalmente é '`/tmp`' ou '`/usr/tmp`'. Se o sistema de arquivo contendo o seu diretório de arquivo temporário é muito pequeno, você deve editar o `mysqld_safe` para configurar `TMPDIR` para apontar para um diretório onde você tenha espaço suficiente! Você também pode definir o diretório temporário usando a opção `--tmpdir` com `mysqld`.

O MySQL cria todos os arquivos temporários como arquivos ocultos. Isto assegura que os arquivos temporários serão removidos se o `mysqld` for terminado. A desvantagem de usar arquivos ocultos é que você não verá um arquivo temporário grande que enche o sistema de arquivos no qual o diretório de arquivos temporários está localizado.

Ao ordenar (`ORDER BY` ou `GROUP BY`), o MySQL normalmente usa um ou dois arquivos temporários. O espaço em disco máximo que você precisa é:

```
(tamanho do que é ordenado + sizeof(apontador do banco de dados))
* números de linhas encontradas
* 2
```

`sizeof(apontados do banco de dados)` normalmene é 4, mas pode crescer no futuro para tabelas realmente grandes.

Para algumas consultas `SELECT`, o MySQL também cria tabelas SQL temporárias. Elas não são ocultas e têm nomes da forma '`SQL_*`'.

`ALTER TABLE` cria uam tabela temporária no mesmo diretório da tabela original.

Se você está usando o MySQL 4.1 ou posterior você pode espalhar a carga entre vários discos físicos definindo `--tmpdir` com uma lista de caminhos separados por dois pontos `:` (ponto e vírgula `;` no Windows). Eles serão feitos através de escalonamento round-robin. **Nota:** Estes caminhos devem ser de diferentes discos **físicos**, e não partições diferentes do mesmo disco.

A.4.5 Como Proteger ou AlterarHow to Protect or Change the MySQL Socket File '`/tmp/mysql.sock`'

Se você tiver problemas com o fato que de que qualquer um pode deletar o socket de comunicação '`/tmp/mysql.sock`' do MySQL, você pode, na maioria das versões Unix, protejer o seu sistema de arquivos '`/tmp`' definindo o bit `sticky`. Conecte como `root` e faça o seguinte:

```
shell> chmod +t /tmp
```

Isto protejerá o seu sistema de arquivos '`/tmp`' para que os arquivos só possam ser deletados pelo seus donos ou pelo superusuário (`root`).

Você pode verificar se o bit `sticky` está setado executando `ls -ld /tmp`. Se o último bit de permissão é `t`, o bit está configurado

Você pode alterar o local onde o MySQL usa/coloca o arquivo de socket da seguinte maneira:

- Especifique o caminho em uma arquivo de opção local ou global. Por exemplo, coloque em `/etc/my.cnf`:

```
[client]
socket=path-for-socket-file

[mysqld]
socket=path-for-socket-file
```

Veja Seção 4.1.2 [Arquivo de opções], Página 216.

- Especificando isto na linha de comando para o `mysqld_safe` e na maioria dos clientes com a opção `--socket=path-for-socket-file`.
- Especifique o caminho para o socket na variável de ambiente `MYSQL_UNIX_PORT`.
- Definindo o caminho com a opção `--with-unix-socket-path=path-for-socket-file` do `configure`. Veja Seção 2.3.3 [configure options], Página 98.

Você pode testar se o socket funciona com o seguinte comando:

```
shell> mysqladmin --socket=/path/to/socket version
```

A.4.6 Problemas Com Fuso Horário

Se você tiver problema com `SELECT NOW()` retornando valores em GMT e não em sua hora local, você terá que definir a variável de ambinte `TZ` com a seu fuso horário atual. Isto deve ser feito no ambiente no qual o servidor é executado, por exemplo, em `mysqld_safe` ou `mysql.server`. Veja Apêndice F [Variáveis de ambiente], Página 1092.

A.5 Assuntos Relacionados a Consultas

A.5.1 Caso-Sensitivito em Pesquisas

Por padrão, as pesquisas no MySQL são caso-insensitivo (a menos que haja algum conjunto de caracter que nunca seja caso-insensitivo, com `czech`). Isto significa que se você buscar com `nome_coluna LIKE 'a%'`, você obterá todos os valores de colunas que iniciam com `A` ou `a`. Se você quiser fazer esta busca caso-sensitivo, use algo como `INSTR(nome_coluna, "A")=1` para verificar o prefixo. Ou use `STRCMP(nome_coluna, "A") = 0` se o valor da coluna deve se exatamente `"A"`.

Operações de comparações simples (`>=, >, = , < , <=`, ordenando e agrupando) são basedos em cada "valor de ordenação" do caracter. Caracteres com o mesmo valor de ordenação (como 'E', 'e' e 'é') são tratados como o mesmo caracter!

Em versões antigas do MySQL, comparações com `LIKE` eram feitas com o valor de letra maiúscula de cada caracter (E == e mas E <> é). Nas versões mais novas, `LIKE` funciona assim como os outros operadores de comparação.

Se você quiser que uma coluna sempre seja tratada de modo caso-sensitivo, declare a como `BINARY`. Veja Seção 6.5.3 [`CREATE TABLE`], Página 598.

Se você está usando caracteres Chineses na codificação big5, você pode tornar todas as colunas de caracteres `BINARY`. Isto funciona porque a ordenação de caracteres de codificação big5 é baseada na ordem do código ASCII.

A.5.2 Problemas Usando Colunas DATE

O formato de um valor DATE é 'YYYY-MM-DD'. De acordo com o padrão SQL, nenhum outro formato é permitido. Você deve usar este formato em expressões UPDATE e na cláusula WHERE de insrtruções SELECT. Por exemplo:

```
mysql> SELECT * FROM nome_tabela WHERE date >= '1997-05-05';
```

Por conveniência, o MySQL converte automaticamente uma data em um número se a data é usada em um contexto numérico (e vice versa). Ele também é esperto o bastante para permitir uma forma de string "relaxada" em uma atualização e em uma cláusula WHERE que compara uma data a uma coluna TIMESTAMP, DATE, ou DATETIME. (Forma relaxada significa que qualquer caracter de pontuação pode seu usado como separador entre as partes. Por exemplo, '1998-08-15' e '1998#08#15' são equivalentes). O MySQL também pode converter uma string sem separadores (como '19980815'), desde que ela faça sentido como uma data.

A data especial '0000-00-00' pode ser armazenada e recuperada como '0000-00-00'. Ao usar uma data '0000-00-00' com o MyODBC, ele a converterá automaticamente em NULL em sua versão 2.50.12 e acima, porqie o ODBC não pode tratar este tipo de data.

Como o MySQL realiza a conversão descrita acima, a seguinte instrução funcionará:

```
mysql> INSERT INTO nome_tabela (idate) VALUES (19970505);
mysql> INSERT INTO nome_tabela (idate) VALUES ('19970505');
mysql> INSERT INTO nome_tabela (idate) VALUES ('97-05-05');
mysql> INSERT INTO nome_tabela (idate) VALUES ('1997.05.05');
mysql> INSERT INTO nome_tabela (idate) VALUES ('1997 05 05');
mysql> INSERT INTO nome_tabela (idate) VALUES ('0000-00-00');

mysql> SELECT idate FROM nome_tabela WHERE idate >= '1997-05-05';
mysql> SELECT idate FROM nome_tabela WHERE idate >= 19970505;
mysql> SELECT MOD(idate,100) FROM nome_tabela WHERE idate >= 19970505;
mysql> SELECT idate FROM nome_tabela WHERE idate >= '19970505';
```

No entatnto o seguinte não funcionará:

```
mysql> SELECT idate FROM nome_tabela WHERE STRCMP(idate,'19970505')=0;
```

STRCMP() é uma função string, assim ela converte idate em uma string e realiza um comparação de string. Ela não converte '19970505' em uma datae e realiza uma comparaçãas de data.

Note que o MySQL faz uma verificação muito limitada da validade da data. Se você aramazenar uma data incorreto, tal como '1998-2-31', a data invalida será armazenada.

Como o MySQL empacota a data para armazenamento, ele não pode armazenar qualquer data dada como já que ela não caberia dentro do buffer de resultado. As regras de aceitação das datas são:

- Se o MySQL pode armazenar e recuperar um data dada, a data errada é acieta para colunas DATE e DATETIME.
- Todos os valores de dia entre 0-31 são aceitos para qualquer data. Isto torna muito conveniente para plicações web nas quais você pede ano, mês e dia em 3 campos diferentes.

- O campo do dia ou mês pode ser zero. Isto é conveniente se você quiser armazenar uma data de aniversário em uma coluna `DATE` e você não sabea parte da data.

Se a data não pode ser convertida para qualquer valor razoável, um `0` é armazenado no campo `DATE`, o qual será recuperado como `0000-00-00`. Isto é uma questão tanto de velocidade quanto de conveniência já que acreditamos que a responsabilidade do banco de dados é recuperar a mesma data que você armazenou (mesmo se a data não era logicamente correta em todos os casos). Nós pensamos que é papel da aplicação verificar as datas, e não do servidor.

A.5.3 Problemas com Valores `NULL`

O conceito do valor `NULL` é uma fonte comum de confusão para os iniciantes em SQL, que frequentemente pensa que `NULL` é a mesma coisa que uma string vazia `""`. Este não é o caso! Por exemplo, as seguintes intruções são completamente diferentes:

```
mysql> INSERT INTO minha_tabela (telefone) VALUES (NULL);
mysql> INSERT INTO minha_tabela (telefone) VALUES ("");
```

Ambas as intruções inserem um valor na coluna `telefone`, mas a primeira insere um valor `NULL` e a segunda insere uma string vazia. O significado do primeiro pode ser considerado como "telefone não é conhecido" e o significado da segunda pode ser considerado como "ela não tem telefone".

Em SQL, o valor `NULL` é sempre falso em coparação a qualquer outro valor, mesmo `NULL`. Uma expressão que contém `NULL` sempre produz um valor `NULL` a menos que seja indicado na documentação para os operadores e funções involvidos na expressão. Todas as colunas no seguinte exemplo retornam `NULL`:

```
mysql> SELECT NULL,1+NULL,CONCAT('Invisible',NULL);
```

Se você quiser procurar por uma coluna cujo valor é `NULL`, você nãp pode usar o teste `=NULL`. A seguinte instrução não retorna nenhuma linha, pois `expr = NULL` é FALSO, para qualquer expressão:

```
mysql> SELECT * FROM minha_tabala WHERE phone = NULL;
```

Para procurar por valores `NULL`, você deve usar o teste `IS NULL`. A seguir mostramos como encontrar o némuro de telefone `NULL` e o número de telefone vazio:

```
mysql> SELECT * FROM minha_tabela WHERE telefone IS NULL;
mysql> SELECT * FROM minha_tabela WHERE telefone = "";
```

Note que você pode adicionar um índice a uma coluna que tenha valores `NULL` apenas se você estiver usando o MySQL versão 3.23.2 ou mais novo e estiver usando tipos de tabelas `NyISAM`, `InnoDB` ou `BDB`. Em versões anteriores e com outros tipos de tabelas, você deve declara tais colunas como `NOT NULL`. Isto também significa que você então não poderá inserir `NULL` em uma coluna indexada.

Ao ler dados com `LOAD DATA INFILE`, colunas vazias são atualizadas com `''`. Se você quiser um valor `NULL` em uma coluna, você deve usar `\N` no arquivo texto. A palavra literal `'NULL'` também pode ser usada em algumas circunstâncias. Veja Seção 6.4.8 [`LOAD DATA`], Página 588.

Ao usar `ORDER BY`, valores `NULL` são apresentados primeiro, ou por último se você especificar `DESC` para armazenar em ordem decrescente. Exceção: Nos MySQL 4.0.2 até 4.0.10, se você armazenar em ordem decrescente usando `DESC`, valores `NULL` são apresentados por último.

Ao usar `GROUP BY`, todos os valores `NULL` são considerados iguais.

Funções de agrupamento (resumo) como `COUNT()`, `MIN()` e `SUM()` ignoram valores `NULL`. A exceção a isto é `COUNT(*)`, que conta linhas e não colunas individuais. Por exemplo, a seguinte instrução deve produzir duas contagens. A primeira é a contagem do número de linhas na tabela e a segunda é a contagem do número de valores diferentes de `NULL` na coluna `age`:

```
mysql> SELECT COUNT(*), COUNT(age) FROM person;
```

Para ajudar com o tratamento de `NULL`, você pode usar os operadores `IS NULL` e `IS NOT NULL` e a função `IFNULL()`.

Para alguns tipos de colunas, valores `NULL` são tratados de forma especial, Se você inserir `NULL` na primeira coluna `TIMESTAMP` de uma tabela, a data e hora atual serão inseridos. Se você isere `NULL` em uma coluna `AUTO_INCREMENT`, o próximo número na sequência é inserida.

A.5.4 Problemas com `alias`

Você pode usar um alias para referir a uma coluna no `GROUP BY`, `ORDER BY`, ou na parte `HAVING`. Aliases podem ser usados para dar as colunas nomes melhores:

```
SELECT SQRT(a*b) as rt FROM nome_tabela GROUP BY rt HAVING rt > 0;
SELECT id,COUNT(*) AS cnt FROM nome_tabela GROUP BY id HAVING cnt > 0;
SELECT id AS "Customer identity" FROM nome_tabela;
```

Note que o padrão SQL não permite que você se refira a uma alias na cláusula `WHERE`. Isto é porque quando o código `WHERE` é executado o valor da coluna ainda não pode ser determinado. Por exemplo, a seguinte consulta é **ilegal**:

```
SELECT id,COUNT(*) AS cnt FROM nome_tabela WHERE cnt > 0 GROUP BY id;
```

A instrução `WHERE` é executada para determinar quais linhas devem ser incluídas na parte `GROUP BY` enquanto `HAVING` é usado para decidir quais linhas o conjunto de resultados deve usar.

A.5.5 Deletando Linhas de Tabelas Relacionadas

Como o MySQL não suporta subconsultas (antes da versão 4.1), enm o uso de mais de uma tabela na instruçao `DELETE` (antes da versão 4.0), você deve usar a seguinte abordagem para deletar linhas de 2 tabelas relacionadas:

1. `SELECT` as linhas baseado em alguma condição `WHERE` na tabela principal.
2. `DELETE` as linhas da tabela princiapl basada nas mesmas condições.
3. `DELETE FROM tabela_relacionada WHERE coluna_relacionada IN (linhas_selecionadas)`.

Se o número total de caracteres na consulta com `colunas_relacionadas` é maior que 1,048,576 (o valor padrão de `max_allowed_packet`, você deve separá-lo em duas partes menores e executar múltiplas instruções `DELETE`. Você provavelmente obterá o `DELETE` mais

rápido apenas delatando 100-1000 ids de `colunas_relacionadas` por consulta se `colunas_relacionadas` é um índice. Se `colunas_relacionadas` não é um índice, a velocidadi é independente do número de argumentos na cláusula `IN`.

A.5.6 Resolvendo Problemas Com Registros Não Encontrados

If you have a complicated query that has many tables and that doesn't return any rows, you should use the following procedure to find out what is wrong with your query:

1. Teste a consulta com `EXPLAIN` e verifique se você pode encontrar alguma coisa que está errada. Veja Seção 5.2.1 [`EXPLAIN`], Página 424.
2. Selcione apenas aqueles campos que são usados na cláusula `WHERE`.
3. Remova uma tabela por vez da consulta até que ela retorne alguns registros. Se as tabelas são grandes, é uma boa idéia usar `LIMIT 10` com a consulta.
4. Faça um `SELECT` da coluna encontrou um registro com a tabela que foi removido por última da consulta.
5. Se você estiver comparando colunas `FLOAT` ou `DOUBLE` com números que tenham decimais, você não pode usar `'='`. Este problema é comum na maioria das linguagens de computadores porque valores de ponto-flutuante não são valores exatos. Na maioria dos casos, alterar o `FLOAT` por `DOUBLE` corrigirá isto. Veja Seção A.5.7 [Problemas com float], Página 934.
6. Se você ainda não pode imaginar o que está errado, crie um teste mínimo que possa ser executado com `mysql test < query.sql` e possa mostrar seus problemas. Você pode criar um arquivo de teste com `mysqldump --quick banco_de_dados tabela > query.sql`. Abra o arquivo em um editor, remova algumas linhas inseridas (se houver muitas) e adicione sua instrução select no fim do arquivo.

   Teste se você ainda está tendo problemas fazendo:

   ```
   shell> mysqladmin create test2
   shell> mysql test2 < query.sql
   ```

   Envie o arquivo de teste usando `mysqlbug` para lista de email gerais do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33.

A.5.7 Problemas com Comparação de Ponto Flutuante

Números de ponto flutuante geram confusões algumas vezes, pois estes números não são armazenados como valores exatos dentro da arquitetura dos computadores. O que pode ser ver na tela não é o valor exato do número.

Tipos de campos `FLOAT`, `DOUBLE` e `DECIMAL` são assim.

```
CREATE TABLE t1 (i INT, d1 DECIMAL(9,2), d2 DECIMAL(9,2));
INSERT INTO t1 VALUES (1, 101.40, 21.40), (1, -80.00, 0.00),
(2, 0.00, 0.00), (2, -13.20, 0.00), (2, 59.60, 46.40),
(2, 30.40, 30.40), (3, 37.00, 7.40), (3, -29.60, 0.00),
(4, 60.00, 15.40), (4, -10.60, 0.00), (4, -34.00, 0.00),
(5, 33.00, 0.00), (5, -25.80, 0.00), (5, 0.00, 7.20),
(6, 0.00, 0.00), (6, -51.40, 0.00);
```

```
mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+-------+
| i    | a      | b     |
+------+--------+-------+
|    1 |  21.40 | 21.40 |
|    2 |  76.80 | 76.80 |
|    3 |   7.40 |  7.40 |
|    4 |  15.40 | 15.40 |
|    5 |   7.20 |  7.20 |
|    6 | -51.40 |  0.00 |
+------+--------+-------+
```

O resultado está correto. Embora pareça que os primeiros cinco registros não devessem passar no teste de comparação, eles deviam porque a diferença entre o número mostrado está na décima casa decimal ou depende da arquitetura do computador.

O problema não pode ser resolvido usando ROUND() (ou função similar), porque o resultado ainda é um número de ponto flutuante. Exemplo:

```
mysql> SELECT i, ROUND(SUM(d1), 2) AS a, ROUND(SUM(d2), 2) AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+-------+
| i    | a      | b     |
+------+--------+-------+
|    1 |  21.40 | 21.40 |
|    2 |  76.80 | 76.80 |
|    3 |   7.40 |  7.40 |
|    4 |  15.40 | 15.40 |
|    5 |   7.20 |  7.20 |
|    6 | -51.40 |  0.00 |
+------+--------+-------+
```

É assim que o número da coluna 'a' se parece:

```
mysql> SELECT i, ROUND(SUM(d1), 2)*1.0000000000000000 AS a,
    -> ROUND(SUM(d2), 2) AS b FROM t1 GROUP BY i HAVING a <> b;
+------+----------------------+-------+
| i    | a                    | b     |
+------+----------------------+-------+
|    1 |  21.3999999999999986 | 21.40 |
|    2 |  76.7999999999999972 | 76.80 |
|    3 |   7.4000000000000004 |  7.40 |
|    4 |  15.4000000000000004 | 15.40 |
|    5 |   7.2000000000000002 |  7.20 |
|    6 | -51.3999999999999986 |  0.00 |
+------+----------------------+-------+
```

Dependendo da arquitetura do computador você pode ou não ver resultados similares. Cada CPU pode avaliar um número de ponto flutuante de forma diferente. Por exemplo, em

alguma máquinas você pode obter resultados 'corretos' multiplicando ambos argumentos por 1, como no exemplo a seguir.

AVISO: NUNCA CONFIE NESTE MÉTODO EM SUAS APLICAÇÕES, ESTE É UM EXEMPLO DE UM MÉTODO ERRADO!!!

```
mysql> SELECT i, ROUND(SUM(d1), 2)*1 AS a, ROUND(SUM(d2), 2)*1 AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+------+
| i    | a      | b    |
+------+--------+------+
|    6 | -51.40 | 0.00 |
+------+--------+------+
```

A razão pela qual o método acima parece funcionar é que na máquina onde o teste foi realizado, a CPU de aritimética de ponto flutuante é realizada arredondando números para serem iguais, mas não há nenhuma regra que qualquer CPU deva fazer assim, então isto não é confiável.

O modo correto de fazermos comparações de ponto flutuante é primeiro decidir qual é a tolerância desejada entre os números e então fazer a comparação com o número tolerado. Por exemplo, se nós concordarmos que números de ponto flutuante devem ser considerados o mesmo, se eles forem o mesmo com precisão de quatro casas deciamis (0.0001), a comparação deve ser feita assim:

```
mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a - b) > 0.0001;
+------+--------+------+
| i    | a      | b    |
+------+--------+------+
|    6 | -51.40 | 0.00 |
+------+--------+------+
1 row in set (0.00 sec)
```

E vice-versa, se nós quisermos obter registros onde os números são o mesmo, o teste seria:

```
mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a - b) < 0.0001;
+------+-------+-------+
| i    | a     | b     |
+------+-------+-------+
|    1 | 21.40 | 21.40 |
|    2 | 76.80 | 76.80 |
|    3 |  7.40 |  7.40 |
|    4 | 15.40 | 15.40 |
|    5 |  7.20 |  7.20 |
+------+-------+-------+
```

A.6 Assuntos Relacionados ao Otimizador

O MySQL usa um otimizador baseado no custo para descobrir o melhor modo de resolver uma consulta. Em muitos casos o MySQL pode calcular a melhor consulta possível mas em

alguns casos o MySQL não tem informação suficiente sobre os dados e precisa fazer alguns palpites sobre os dados.

Esta seção do manual é direcionada para os casos nos quais o MySQL não faz isto corretamente.

A ferramenta que se tem disponível para ajudar o MySQL a fazer as coisas 'certas' são:

- `EXPLAIN`. Veja Seção 5.2.1 [`EXPLAIN`], Página 424.
- `ANALYZE TABLE`. Veja Seção 4.6.2 [`ANALYZE TABLE`], Página 298.
- `USE INDEX`, `FORCE INDEX` and `IGNORE INDEX`. Veja Seção 6.4.1 [SELECT], Página 562.
- `STRAIGHT JOIN` a nível de tabela e global. Veja Seção 6.4.1 [SELECT], Página 562.
- Configurar variáveis específicas de threads. Veja Seção 4.6.8.4 [SHOW VARIABLES], Página 309.

A.6.1 Camo evitar o varredura da tabela,,,

`EXPLAIN` mostrará `ALL` na coluna `type` quando o MySQL usa uma busca na tabela para resolver uma consulta. Isto acontece normalmente quando:

- A tabela é tão pequena que é mais rápido fazer uma varredura na tabela que uma busca nas chaves. Isto é um caso comum para tabelas com menos de 10 linhas e um tamanho de linha pequeno.
- Não há nenhum restrição utilizável na cláusula `ON` ou `WHERE` para colunas indexadas.
- Você está comparando colunas indexadas com constantes e o MySQL calculou (baseado na árvore de índices) que a constante cobre uma parte muito grande da tabela e uma busca na tabela seria mais rápido.. Veja Seção 5.2.4 [Where optimizations], Página 432.
- Você está usando uma chave com baixa cardinalidade (= muitos registros coincidentes) através de outra coluna. O MySQL assumirá neste caso que usar a chave fará muitas pesquisas de chave e neste caso a varredura da tabela seria mais rápido.

O que você pode fazer para evita uma busca 'errada' em tabelas grandes é:

- Use `ANALYZE TABLE` para a tabela em quastão atualizar a distribuição das chaves.. Veja Seção 4.6.2 [ANALYZE TABLE], Página 298.
- Use `FORCE INDEX` para a tabela em questão para dizer ao MySQL que uma busca na tabela é muito cara comparado com usar um dos índices dados. Veja Seção 6.4.1 [SELECT], Página 562.

```
SELECT * FROM t1,t2 force index(index_for_column) WHERE t1.column=t2.column;
```

- Inicie o `mysqld` com `--max-seeks-for-key=1000` ou faça `SET MAX_SEEKS_FOR_KEY=1000` para dizer ao otimizador que nenhuma busca de chave fará mais que 1000 pesquisas nas chaves.

A.7 Assuntos Relacionados a Definições de Tabelas

A.7.1 Problemas com `ALTER TABLE`.

`ALTER TABLE` altera uma tablea para o conjunto de caracteres atual. Se você obter um erro de chave duplicada durante `ALTER TABLE`, então a causa é que o novo conjunto de caracteres mapeia duas chaves para o mesmo valor ou que a tabela está corrompida, caso no qual você deve fazer um `REPAIR TABLE` na tabela.

Se `ALTER TABLE` finalizar com um erro com este:

```
Error on rename of './database/name.frm' to './database/B-a.frm' (Errcode: 17)
```

o problema pode ser que o MySQL falhou em um `ALTER TABLE` anterior e existe uma tabela antiga chamada '`A-algumacoisa`' ou '`B-algumacoisa`'. Neste caso, vá até o diretório de dados do MySQL e delete todos os campos que tenham nomes iniciando com `A-` ou `B-`. (Você pode quere movê-los para algum lugar em vez de deletá-los.)

`ALTER TABLE` funciona do seguinte modo:

- Cria uma nova tabela chamada '`A-xxx`' com as alterações pedidas.
- Todos os registros da tabela antiga são copiadas para '`A-xxx`'.
- A tabela antiga é renomeada com '`B-xxx`'.
- '`A-xxx`' é renomeada com o nome da sua tabela antiga.
- '`B-xxx`' é deletada.

Se algo der errado com a operação de renomeação, o MySQL tenta desfazer a mudança. Se algo der seriamente errado (isto não deve acontecer, é claro), o MySQL pode deixar a tabela antiga como '`B-xxx`', mas uma simples renomeação no nível do sistema deve trazer o seus dados de volta.

A.7.2 Como Alterar a Ordem das Colunas em Uma Tabela

O ponto principal do MySQL é abstrair a aplicação do formato de armazenamento dos dados. Você sempre deve especificar a ordem na qual você deseja recuperar os dados. Por exemplo:

```
SELECT nome_coluna1, nome_coluna2, nome_coluna3 FROM nome_tabela;
```

retornará na ordem `nome_coluna1`, `nome_coluna2`, `nome_coluna3`, enquanto:

```
SELECT nome_coluna1, nome_coluna3, nome_coluna2 FROM nome_tabela;
```

retornará colunas na ordem `nome_coluna1`, `nome_coluna3`, `nome_coluna2`.

Se você quiser alterar a ordem das colunas, você pode fazer o seguinte:

1. Crie uma nova abela com as colunas na ordem correta.
2. Execute `INSERT INTO tabela_nova SELECT campos-na-ordem-de-tabela_nova FROM tabela_antiga`.
3. Delete ou renomeie `tabela_antiga`.
4. `ALTER TABLE tabela_nova RENAME tabela_antiga`.

Em uma aplicação, você **nunca** deve usar `SELECT *` e recuperar as colunas baseado em suas posições, pois a ordem e a posição nas quais as colunas são retornadas não permanecerá a mesma se você adicionar/mover/deletar colunas. Uma simples alteração na estrutura de seu banco de dados causaria uma falha em sua aplicação. É claro que `SELECT *` é muito mais cabível em testes de cosultas.

A.7.3 Problemas com TEMPORARY TABLE

Segue uma lista de limitações com `TEMPORARY TABLES`.

- Uma tabela temporária só pode ser do tipo `HEAP`, `ISAM`, `MyISAM`, `MERGE`, ou `InnoDB`.
- Você não pode usar tabelas temporárias mais que uma vez na mesma consulta. Por exemplo, o seguinte não funciona.

```
mysql> SELECT * FROM tabela_temporária, tabela_temporária AS t2;
```

- Você não pode usar `RENAME` em uma tabela temporária (`TEMPORARY`). Note que `ALTER TABLE nome_orig RENAME nome_novo` funciona!

Apêndice B Contribuição de Programas

Muitos usuários do MySQL têm contribuído com *muitas* ferramentas de suporte e add-ons úteis.

Uma lista de alguns programas disponíveis no website do MySQL (ou qualquer) mirror é apresentada aqui.

Você também pode visitar nosso nossa lista online de programas relacionados ao MySQL em `http://www.mysql.com/portal/software/`. As facilidades da comunidade também permitem suas contribuições!

Se você quiser construir suporte ao MySQL para interface Perl `DBI`/ `DBD`, você deve buscar os arquivos `Data-Dumper`, `DBI` e `DBD-mysql` e instalá-los. Veja Seção 2.7 [Suporte Perl], Página 164.

Note: Os programas listados aqui podem ser baixados e usados livremente. Os direitos autorias pertencem aos seus respectivos donos. Por favor, veja a documentação de cada produto para maiores detalhes sobre licenciamento e termos. A MySQL AB não assume responsabilidade sobre a validade das informações neste capítulo ou sobre a operação apropriada dos programas listados aqui.

B.1 APIs

- Perl Modules
  - `http://www.mysql.com/Downloads/Contrib/KAMXbase1.2.tar.gz` Converte arquivos '`.dbf`' em tabelas do MySQL. Módulo Perl escrito por Pratap Pereira `pereira@ee.eng.ohio-state.edu`, expandido por Kevin A. McGrail `kmcgrail@digital1.peregrinehw.com`. Este conversor pode tratra campos MEMO.
  - `http://www.mysql.com/Downloads/Contrib/HandySQL-1.1.tar.gz` HandySQL é uma módulo de acesso ao MySQL. Ele oferece uma interface C embutida em Perl e é aproximadamente 20% mais rápida que o DBI regular.
- OLEDB
  - `http://www.mysql.com/Downloads/Win32/MyOLEDB3.exe` Pacote de instalação MyOLEDB 3.0 da SWSoft.
  - `http://www.mysql.com/Downloads/Win32/mysql-oledb-3.0.0.zip` Fonte do MyOLEDB 3.0.
  - `http://www.mysql.com/Downloads/Win32/MySamples.zip` Exemplos e documentação do MyOLEDB.
  - `http://www.mysql.com/Downloads/Win32/MyOLEDB.chm` Arquivos de ajuda do MyOLEDB.
  - `http://www.mysql.com/Downloads/Win32/libmyodbc.zip` Biblioteca estática do MyODBC usada para construir o MyOLEDB. Baseada no código MyODBC.
- C++
  - `http://www.mysql.com/Downloads/Contrib/mysql-c++-0.02.tar.gz` Biblioteca wrapper C++ dp MySQL. Por Roland Haenel, `rh@ginster.net`.

  - `http://www.mysql.com/Downloads/Contrib/MyDAO.tar.gz` API C++ do MySQL. Por Satish `spitfire@pn3.vsnl.net.in`. Inspirado pela API C++ de Roland Haenel e pela biblioteca MyC de Ed Carp.
  - `http://www.mysql.com/products/mysql++/` API C++ do MySQL (mais que uma bibliteca wrapper). Originalmente criado por `kevina@clark.net`. Mantido por Sinisa na MySQL AB.
  - `http://nelsonjr.homepage.com/NJrAPI/` Um biblioteca independente de banco de dados em C++ que suporta MySQL.

- Delphi
  - `http://www.mysql.com/Downloads/Contrib/DelphiMySQL2.zip` Interface Delphi para `libmysql.dll`, por `bsilva@umesd.k12.or.us`.
  - `http://www.mysql.com/Downloads/Contrib/Udmysql.pas` Um wrapper para libmysql.dll para uso com Delphi. Por Reiner Sombrowsky.
  - `http://www.fichtner.net/delphi/mysql.delphi.phtml` Uma interface Delphi para o MySQL, com código fonte. Por Matthias Fichtner.
  - `http://www.productivity.org/projects/tmysql/` `TmySQL`, uma biblioteca para utilizar o MySQL com Delphi.
  - `https://sourceforge.net/projects/zeoslib/` Zeos Library é um conjunto de componentes de banco de dados para MySQL, PostgreSQL, Interbase, MS SQL, Oracle e DB/2. Também inclui ferrementas de desenvolvimento tais como Database Explorer e Database Designer.
  - `http://www.mysql.com/Downloads/Contrib/JdmMysqlDriver-0.1.0.tar.gz` Um driver da VisualWorks 3.0 Smalltalk para MySQL. Por `joshmiller@earthlink.net`.
  - `http://www.mysql.com/Downloads/Contrib/Db.py` Módulo python com caching. Por `gandalf@rosmail.com`.
  - `http://www.mysql.com/Downloads/Contrib/MySQLmodule-1.4.tar.gz` Interface python para MySQL. Por Joseph Skinner `joe@earthlight.co.nz`. Modificado por Joerg Senekowitsch `senekow@ibm.net`.
  - `http://www.mysql.com/Downloads/Contrib/mysqltcl-1.53.tar.gz` Interface Tcl para o MySQL. Baseado no '`msqltcl-1.50.tar.gz`'. Para a versão 2.0 e mais informações, veja `http://www.xdobry.de/mysqltcl/`.
  - `http://www.mysql.com/Downloads/Contrib/MyC-0.1.tar.gz` Uma API com Visual Basic por Ed Carp.
  - `http://www.mysql.com/Downloads/Contrib/Vdb-dflts-2.1.tar.gz` Esta é uma nova versão de um conjunto de bibliotecas utilitárias com a intenção de fornecer uma interface genérica para o mecanismo de banco de dados SQL para que sua aplicação se torne de 3 camadas. A vantagem é que você pode facilmente trocar o mecanismo de banco de dados implementando um arquvo para o novo backend sem qualquer alteração em sua aplicação. Por `damian@cablenet.net`.
  - `http://www.mysql.com/Downloads/Contrib/DbFramework-1.10.tar.gz` DbFramework é uma coleção de classes para manipular o banco de dados MySQL. As classes são baseadas no CDIF Data Model Subject Area. Por Paul Sharpe `paul@miraclefish.com`.

- `http://www.mysql.com/Downloads/Contrib/pike-mysql-1.4.tar.gz` Modulo MySQL para pike. Para uso com o servidor web Roxen.
- `http://www.mysql.com/Downloads/Contrib/squile.tar.gz` Módulo para `guile` que permite ao `guile` interagir com banco de dados SQL. Por Hal Roberts.
- `http://www.mysql.com/Downloads/Contrib/stk-mysql.tar.gz` Interface para o Stk. Stk é o Tk widgets com Scheme em vez do Tcl. Por Terry Jones.
- `http://www.mysql.com/Downloads/Contrib/eiffel-wrapper-1.0.tar.gz` Eiffel wrapper por Michael Ravits.
- `http://www.mysql.com/Downloads/Contrib/SQLmy0.06.tgz` FlagShip Replaceable Database Driver (RDD) para MySQL. Por Alejandro Fernandez Herrero. A homepage do Flagship RDD é `http://www.fship.com/rdds.html`.
- `http://www.mysql.com/Downloads/Contrib/mydsn-1.0.zip` Binário e Fonte para `mydsn.dll`. mydsn deve ser usado para construir e remover o arquivo de registro DSN para o driver MyODBC am aplicações Coldfusion. Por Miguel Angel Solórzano.
- `http://www.mysql.com/Downloads/Contrib/MySQL-ADA95_API.zip` Uma interface ADA95 para a API do MySQL. Por Francois Fabien.
- `http://www.mysql.com/Downloads/Contrib/MyTool-DLL_for_VB_and_MySQL.zip` Uma DLL com a API C do MySQL para Visual Basic. Por Ken Menzel `kenm@icarz.com`.
- `http://www.mysql.com/Downloads/Contrib/MYSQLX.EXE` Objeto ActiveX do MySQL para acesso direto do ser servidor MySQL através do IIS/ASP, VB, VC++ evitando o ODBC que é mais lento. Totalmente atualizável, multi-thread com suporte total para todos os tipos de campos do MySQL (versão 2001.1.1). Por SciBit `http://www.scibit.com/`.
- `http://www.fastflow.it/mylua/` Site do MyLUA; como utilizar a linguagem LUA para escrever `PROCEDURE` MySQL que podem ser carregados em tempo de execução.
  - `http://www.mysql.com/Downloads/Contrib/lua-4.0.tar.gz` LUA 4.0
  - `http://www.mysql.com/Downloads/Contrib/mylua-3.23.32.1.tar.gz` Correção para o MySQL 3.23.32 para usar o LUA 4.0. Por Cristian Giussani.
- `http://www.mysql.com/Downloads/Contrib/patched_myodbc.zip` Correção (para suporte ao Omniform 4.0) para o driver MyODBC. Por Thomas Thaele `tthaele@papenmeier.de`

B.2 Conversores

- `http://www.mysql.com/Downloads/Contrib/mssql2mysql.txt` Conversor do MS-SQL para MySQL. Por Michael Kofler. O site do mssql2mysql é `http://www.kofler.cc/mysql/mssql2mysql.html`.
- `http://www.mysql.com/Downloads/Contrib/dbf2mysql-1.14.tar.gz` Conversor de aruivos '`.dbf`' em tabelas MySQL. Por Maarten Boekhold (`boekhold@cindy.et.tudelft.nl`), William Volkman, e Michael Widenius. Este conversor inclui suporte rudimentar a campos MEMO para somente-leitura.

- `http://www.mysql.com/Downloads/Contrib/dbf2mysql-1.13.tgz` Converte arquivos '`.dbf`' em tabelas MySQL. Por Maarten Boekhold, `boekhold@cindy.et.tudelft.nl`, e Michael Widenius. Este conversor não pode tratar campos MEMO.
- `http://www.mysql.com/Downloads/Contrib/dbf2mysql.zip` Converte arquivos FoxPro '`.dbf`' em tabelas MySQL no Windows. Por Alexander Eltsyn, `ae@nica.ru` ou `ae@usa.net`.
- `http://www.mysql.com/Downloads/Contrib/dbf2sql.zip` Programa pequeno e simples que pode lhe ajudar a transportar os dados de uma tabela foxpro em uma tabela MySQL. Por Danko Josic.
- `http://www.mysql.com/Downloads/Contrib/dump2h-1.20.gz` Converte a saída de um `mysqldump` em um arquivo de cabeçalho (.h) do C. Por Harry Brueckner, `brueckner@mail.respublica.de`.
- `http://www.mysql.com/Downloads/Contrib/exportsql.txt` Um script parecido com `access_to_mysql.txt`, exceto que ele é totalmente configurável, tem melhor conversão de tipo (incluindo detecção de campos `TIMESTAMP`), fornece avisos e sugestões enquanto converte, coloca **todos** os caracteres especiais em dados binários e texto e assim por diante. Também converte para o `mSQL` v1 e v2, e não tem custo. Veja `http://www.cynergi.net/exportsql/` para a última versão. Por Pedro Freire, `support@cynergi.net`. Note: Nào funciona com Access 2.0!
- `http://www.mysql.com/Downloads/Contrib/access_to_mysql.txt` Cole esta função em um módulo Access de um banco de dados que possua as tabelas que você deseja exportar. Veja também o `exportsql`. Por Brian Andrews. Nota: Não funciona com Access 2.0!
- `http://www.mysql.com/Downloads/Contrib/importsql.txt` Um script que faz exatamente o contrário do `exportsql.txt`. Ou seja, ele importa dados do MySQL para um banco de dado no Access via ODBC. Ele é bem acessível quando combinado com exportsql, pois ele deixa você uar o Access para todo desenvolvimento e administração do BD, e sincroniza com o seu servidor MySQL. Sem custos. Veja `http://www.netdive.com/freebies/importsql/` para atualizações. Criado por Laurent Bossavit da NetDIVE. **Nota**: não funciona com Access2!
- `http://www.mysql.com/Downloads/Contrib/mdb2sql.bas` Conversor do Access97 para MySQL por Moshe Gurvich.
- `http://www.mysql.com/Downloads/Contrib/msql2mysqlWrapper-1.0.tgz` Um wrapper C do `mSQL` para MySQL. Por `alfred@sb.net`
- `http://www.mysql.com/Downloads/Contrib/sqlconv.pl` Um script simples que pode ser usado para copiar campos de uma tabela MySQL para outro, em blocos. Basicamente você pode rodar `mysqldump` e canalizá-lo para o script `sqlconv.pl`. O script analisará a saída do `mysqldump` e rearranjará os campos para que sejam inseridosd em uma nova tabela. Um exemplo é quando você quer criar uma nova tabela em local diferente do qual você está trabalhando, mas a tabela é apenas um pouco diferente (isto é - campos em ordem diferente, etc.). Por Steve Shreeve.
- `http://www.mysql.com/Downloads/Contrib/oracledump` Progrma Perl para converter bancos de dados Oracle em MySQL. Tem o mesmo formato de saída do mysqldump. Por Johan Andersson.

- http://www.mysql.com/Downloads/Contrib/excel2mysql Programa Perl para importar pastas de trabalho do Excel em um banco de dados MySQL. Por Stephen Hurd shurd@sk.sympatico.ca
- http://www.mysql.com/Downloads/Contrib/T2S_100.ZIP. Programa Windos para converter arquivos textos em banco de dados MySQL. Por Asaf Azulay.

B.3 Utilitários

- http://worldcommunity.com/opensource/utilities/mysql_backup.html MySQL Backup é um script de backup para o MySQL. Por Peter F. Brown.
- http://www.mysql.com/Downloads/Contrib/mysql_watchdog.pl Monitora a daemon do MySQL par possíveis travamentos. Por Yermo Lamers, yml@yml.com.
- http://www.mysql.com/Downloads/Contrib/mysql_structure_dumper.tar.gz
- http://www.mysql.com/Downloads/Contrib/mysql_structure_dumper.tgz Exibe a estrutura de toda a tabela em um banco de dados. Por Thomas Wana.
- http://www.mysql.com/Downloads/Contrib/mysqlsync. Um script Perl para manter cópias remotas de uma banco de dados MySQL em sioncronia com uma cópia master central. Por Mark Jeftovic. markjr@easydns.com.
- http://www.mysql.com/Downloads/Contrib/MySQLTutor-0.2.tar.gz. MySQL-Tutor. Um tutorial MySQL para iniciantes.
- http://www.mysql.com/Downloads/Contrib/mysql_replicate.pl Programa Perl que trata replicações. Por elble@icculus.nsg.nwu.edu
- http://www.mysql.com/Downloads/Contrib/dbcheck Script Perlque tira backup de uma tabela antes de executar isamchk nelas. Por Elizabeth.
- http://www.mysql.com/Downloads/Contrib/mybackup.
- http://www.mswanson.com/mybackup (home page do mybackup) Wrapper para o mysqldump para backup de todos os bancos de dados. Por Marc Swanson.
- http://www.mysql.com/Downloads/Contrib/mdu.pl.gz Exibe o armazenamento usado em um banco de dados MySQL.

Apêndice C Colaboradores do MySQL

Este apêndice lista o desenvolvedores, coolaboradores e responsáveis por suporte que ajudaram a fazer o MySQL o que ele é hoje.

C.1 Desenvolvedores do MySQL

Estes são os desenvolvedores que que são ou foram empregaos pela `MySQL AB` para trabalhar no programa de banco de dados `MySQL`, listado na ordem em que começaram a trabalhar para nós. Na sequência de cada um dos desenvolvedores está uma pequena lista de tarefas pelas quais o desenvolvedor é resposável ou as realizações de cada um. Todos os desenvolvedores estão envovidos no suporte.

Michael (Monty) Widenius

- Desenvolvedor líder e prioncipal autor do servidor MySQL (`mysqld`).
- Novas funções para a biblioteca de string.
- A maioria das bibliotecas `mysys`.
- As biblotecas `ISAM` e `MyISAM` (tratamento do arquivo de índices em árvore-B e compactação do índice e formato de regitsros diferentes).
- A biblioteca `HEAP`. Um sistema de tabela em memória com nosso hashing totalmente dinâmico. Em uso desde 1981 e publicado em 1984.
- O programa `replace` (gastou bastante tempo nele, é bem **LEGAL**!).
- `MyODBC`, o driver ODBC para Windows95.
- Correção de bugs nas MIT-pthreads para fazê-la funcionar com o Servidor MySQL. E também Unireg, uma ferramenta com muitas utilidades.
- Portabilidade de ferramentas `mSQL` como `msqlperl`, `DBD/DBI`, e `DB2mysql`.
- A maioria dos programas `crash-me` e a fundação do benchmarks do MySQL.

David Axmark

- Principal escritor inicial do **Manual de Referência**, incluindo melhoras no `texi2html`.
- Atualização automatica do manual no site.
- Suporte incial ao Autoconf, Automake, e Libtool.
- Licenciamento.
- Partes de todos os arquivos textos. (Hoje em dia apenas o '`README`' é deixado. O reto é incluído no manual.)
- Varios testes de novos recursos.
- Nosso expert em assuntos legais de Software Livre.
- Responsável pela lista de email (que nunca tem tempo para fazê-lo corretamente...).
- Nossa portabilidade do código original (mais de 10 anos). Hoje em dia apenas algumas partes do `mysys` foram deixadas.

- Alguém para o Monty chamar no meio da noite que ele percebe que aquele novo recurso funciona.
- Chefe "Open Sourcerer" (relações na comunidade MySQL).

Jani Tolonen

- `mysqlimport`
- Diversas extensões dos clientes de linha de comando.
- `PROCEDURE ANALYSE()`

Sinisa Milivojevic

- Compactação (com `zlib`) no protocolo cliente/servidor.
- Hashing perfeito para fase do analisador lexicográfico.
- `INSERT` multi-linhas
- Opção -e do`mysqldump`
- `LOAD DATA LOCAL INFILE`
- Opção `SQL_CALC_FOUND_ROWS` do `SELECT`
- Opção `--max-user-connections=...`
- `net_read` e `net_write_timeout`
- `GRANT/REVOKE` e `SHOW GRANTS FOR`
- Novo protocolo cliente/servidor para 4.0
- `UNION` na versão 4.0
- `DELETE/UPDATE` multi-tabelas
- Tabelas derivadas na versão 4.1
- Gerênciamento de recursos do usuário
- Desenvolvedor inicial da APC C++ `MySQL++` e do cliente `MySQLGUI`.

Tonu Samuel (past developer)

- interface VIO (a fundação para o protocolo cliente/servidor criptografado).
- Sistema de arquivos do MySQL (um modo de usar banco de dados MySQL como arquivos e diretórios).
- A expressão `CASE`.
- As funções `MD5()` e `COALESCE()`.
- Suporte `RAID` para tabelas `MyISAM`.

Sasha Pachev

- Implementação inicial da replicação (até versão 4.0).
- `SHOW CREATE TABLE`.
- `mysql-bench`

Matt Wagner

- Pacote de teste do MySQL
- Webmaster (até 2002).
- Coordenaçdor do desenvolvimento.

Miguel Solorzano

- Desenvolvimento e contrução das distribuições Win32
- Código do servidor Windows NT.
- WinMySQLAdmin

Timothy Smith (past developer)

- Suporte a conjunto de caracteres dinâmicos.
- configure, RPMs e outra partes dos sistemas construídos.
- Desenvolvedor inicial do `libmysqld`, o servidor embutido.

Sergei Golubchik

- Pesquisa Full-text.
- Adioção de chaves à biblioteca `MERGE`.

Jeremy Cole

- Aprovação e edição deste manual.
- `ALTER TABLE ... ORDER BY ...`.
- `UPDATE ... ORDER BY ...`.
- `DELETE ... ORDER BY ...`.

Indrek Siitan

- Design/programação de nossa interface web.
- Autor do nosso sistema de gerenciamento de newsletter.

Jorge del Conde

- `MySQLCC` (`MySQL Control Center`)
- Desenvolvento do Win32
- Implantação inicial do portal na web.

Venu Anuganti

- Connector/ODBC (MyODBC) 3.51
- Novo protocolo cliente/servidor para a versão 4.1 (para instruções preparadas).
-

Arjen Lentz

- Responsável pelo Manual de Referência do MySQL
- Preparação da edição impressa do Manual.

Alexander (Bar) Barkov, Alexey (Holyfoot) Botchkov, and Ramil Kalimullin

- Dados espaciais (GIS) e implementação de Arvores-R para versão 4.1
- Unicode e conjunto de caracteres para versão 4.1; documentação para os mesmos.

Oleksandr (Sanja) Byelkin

- Cache de consultas na versão 4.0
- Implementação de subconsultas (4.1).

Aleksey (Walrus) Kishkin and Alexey (Ranger) Stroganov

- Análise e desenho dos benchmarks.

- Manutenção do pacote de teste do MySQL.

Zak Greant
- Advogado do Open Source, relações da comunidade MySQL

Carsten Pedersen
- O programa de certificação do MySQL.

Lenz Grimmer
- Engenharia de produção (contrução e distribuição)

Peter Zaitsev
- Funções `SHA1()`, `AES_ENCRYPT()` e `AES_DECRYPT()`.
- Depuração, pondo em ordem vários recursos.

Alexander (Salle) Keremidarski
- Suporte.
- Depuração.

Per-Erik Martin
- Desenvolvedor responsável por stored procedures (5.0) e triggers.

Jim Winstead
- Lidera o desenvolvimento web

Mark Matthews
- Driver do Connector/J (Java).

Peter Gulutzan

Adequação aos padrões SQL-99, SQL:2003.
- Documentação do algoritmo/código existente do MySQL.
- Documentação do conjunto de caracteres.

Guilhem Bichot
- Replciação, a partir do `MySQL` versão 4.0.
- Correção do tratamento de expoentes para `DECIMAL`.
- Autor do `mysql_tableinfo`.

Antony T. Curtis
- MySQL Database para OS/2.

C.2 Coolaboradores do MySQL

Enquanto a `MySQL AB` for dona dos direitos autorais do `servidor MySQL` e do `manual MySQL`, desejamos reconhecer aqueles que tiveram contibuições de qualquer tipo na `distribuição do MySQL`. Os colaboradores estão listados aqui, em uma ordem randômica:

Gianmassimo Vigazzola `qwerg@mbox.vol.it` or `qwerg@tin.it`

A portabilidade inicial para Win32/NT.

Per Eric Olsson

Pelas críticas mais ou menos condtrutivas e pelo teste do formato de registro dinâmico.

Irena Pancirov `irena@mail.yacc.it`
: Portabilidade para Win32 com compilador Borland. `mysqlshutdown.exe` e `mysqlwatch.exe`

David J. Hughes
: Pelo esforço para fazer um banco de dados SQL shareware. Na TcX, a predecessora da MySQL AB, iniciamos com `mSQL`, mas achamos que ele não podia satisfazer os nossos propositos assim escrevemos uma interface SQL para nossa aplicação Unireg. Os clientes `mysqladmin` e `mysql` são programas que foram largamente influenciados pelo `mSQL`. Nos esforçamos muito tentando fazer da sintaxe do MySQL um superconjunto do `mSQL`. Muitas das idéias de API eram emprestadas do `mSQL` para tornar fácil de se portar programas livres para o `mSQL` para a API do MySQL. O programa MySQL não contém nenhum código do `mSQL`. Dois arquivos na distribuição ('`client/insert_test.c`' e '`client/select_test.c`') são baseados nos arquivos correspondentes (sem direitos autorais) na distribuição do `mSQL`, mas são modificados como exemplo mostrando as alterações necessárias para converter um código do `mSQL` para o servidor MySQL.. (`mSQL` e de direito autora de David J. Hughes.)

Patrick Lynch
: Por ajudar-nos a adquirir o `http://www.mysql.com/`.

Fred Lindberg
: Por configurar o qmail para tratar a lista de email do MySQL e pela incrível ajuda que obtemos gerenciando a lista de emails do MySQL.

Igor Romanenko `igor@frog.kiev.ua`
: `mysqldump` (antigo `msqldump`, mas portado e aprimorado por Monty).

Yuri Dario
: Por manter e expandir a portabilidade do MySQL para OS/2.

Tim Bunce
: Autor do `mysqlhotcopy`.

Zarko Mocnik `zarko.mocnik@dem.si`
: Ordenação em esloveno.

"TAMITO" `tommy@valley.ne.jp`
: O macro do conjunto de caracteres `_MB` e os conjuntos de caracteres ujis e sjis.

Joshua Chamas `joshua@chamas.com`
: Base para inserções concorrentes, sintaxe da data estendida, depuração no NT resposta na lista de email do MySQL.

Yves Carlier `Yves.Carlier@rug.ac.be`
: `mysqlaccess`, um progrma para mostrar os direitos de acesso do usuário.

Rhys Jones `rhys@wales.com` (e GWE Technologies Limited)
: Por um dos primeiros drives JDBC.

Dr Xiaokun Kelvin ZHU `X.Zhu@brad.ac.uk`
: Desenvolvimento de um dos primeiros drivers JDBC e outras ferramentas Java relacionadas ao MySQL.

James Cooper `pixel@organic.com`
: Por configurar um arquivo de lista de email com busca em seu site.

Rick Mehalick `Rick_Mehalick@i-o.com`
: Pelo `xmysql`, um cliente gráfico X para o servidor MySQL.

Doug Sisk `sisk@wix.com`
: Por fornecer pacotes RPM do MySQL para Linux Red Hat

Diemand Alexander V. `axeld@vial.ethz.ch`
: Por fornecer pacotes RPM do MySQL para Linux Red Hat-Alpha.

Antoni Pamies Olive `toni@readysoft.es`
: Por fornecer versões RPM de vários clientes MySQL para Intel e SPARC.

Jay Bloodworth `jay@pathways.sde.state.sc.us`
: Por forncer versões RPM do MySQL versão 3.21.

David Sacerdote `davids@secnet.com`
: Ideias para verificação segura de nomes de máquinas DNS.

Wei-Jou Chen `jou@nematic.ieo.nctu.edu.tw`
: Algum suporte para caracteres chineses (BIG5).

Wei He `hewei@mail.ied.ac.cn`
: Diversas funcionalidades para o conjunto de casracteres chineses(GBK).

Jan Pazdziora `adelton@fi.muni.cz`
: Oredenação em Tcheco

Zeev Suraski `bourbon@netvision.net.il`
: Formatação de tempo `FROM_UNIXTIME()`, funções `ENCRYPT()` e conseleheiro do `bison`. Membro ativo da lista de email.

Luuk de Boer `luuk@wxs.nl`
: Portado (e extendido) o pacote de benchmark para `DBI/DBD`. Tem sido de grande ajuda com o `crash-me` e benchmarks em execução. Algumas novas funções de data. O script `mysql_setpermissions`.

Alexis Mikhailov `root@medinf.chuvashia.su`
: funções definidas por usuários (UDFs); `CREATE FUNCTION` e `DROP FUNCTION`.

Andreas F. Bobak `bobak@relog.ch`
: A extensão `AGGREGATE` para funçÕes UDF.

Ross Wakelin `R.Wakelin@march.co.uk`
: Ajuda na configuração do InstallDhield para o MySQL-Win32.

Jethro Wright III `jetman@li.net`
: A biblioteca '`libmysql.dll`'.

James Pereria `jpereira@iafrica.com`
: Mysqlmanager, uma ferramenta Win32 GUI para administração do servidor MySQL.

Curt Sampson `cjs@portal.ca`
: Potabilidade de MIT-pthreads para NetBSD/Alpha e NetBSD 1.3/i386.

Martin Ramsch `m.ramsch@computer.org`
: Exemplos no Tutorial MySQL.

Steve Harvey
: Por fazer `mysqlaccess` mais seguro.

Konark IA-64 Centre of Persistent Systems Private Limited
: `http://www.pspl.co.in/konark/`. Ajuda com a portabilidade do servidor MySQL para Win64.

Albert Chin-A-Young.
: Atulização do configure para Tru64, suporte a arquivos grandes e suporte a melhores wrappers TCP.

John Birrell
: Emulacao do `pthread_mutex()` para OS/2.

Benjamin Pflugmann
: Exetnsão de tabelas `MERGE` para tratar `INSERTS`. Membro ativo na lista de emails do MySQL.

Jocelyn Fournier
: Excelente ao mostrar e relatar inumeráveis bugs. (especialmente no código da subconsulta no MySQL 4.1)

Marc Liyanage
: Manutenção dos pacotes do Mac OS X e fornecimento de feedbacks sobre como criar pacotes para Mac OS X.

Robert Rutherford
: Por fornecer informações e feedback sobre o port QNX.

Outros colaboradores, pesquisadores de bug e responsaveis por testes: James H. Thompson, Maurizio Menghini, Wojciech Tryc, Luca Berra, Zarko Mocnik, Wim Bonis, Elmar Haneke, `jehamby@lightside`, `psmith@BayNetworks.com`, `duane@connect.com.au`, Ted Deppner `ted@psyber.com`, Mike Simons, Jaakko Hyvatti.

E vários relatos/correções de bugs do pessoal da lista de email.

Um grande tributo vai àqueles que nos ajudaram a responder dúvidas na lista de email do MySQL.

Daniel Koch `dkoch@amcity.com`
: Configuração do Irix.

Luuk de Boer `luuk@wxs.nl`
: Dúvidas de benchmark.

Tim Sailer `tps@users.buoy.com`
: Questões do `DBD-mysql`.

Boyd Lynn Gerber `gerberb@zenez.com`
: Questões relacionadas ao SCO.

Richard Mehalick `RM186061@shellus.com`
: Questões relacionadas ao `xmysql` e questões básicas de instalação.

Zeev Suraski `bourbon@netvision.net.il`
: Questões de configuração do módulo Apache (log & autent) e questões relacionadas ao PHP, questões relacionadas a sintaxe SQL e outras questões gerais.

Francesc Guasch `frankie@citel.upc.es`
: Questões gerais.

Jonathan J Smith `jsmith@wtp.net`
: Questões específicas do SO Linux, sintaxe SQL e outra coisas que podem precisar de algum trabalho.

David Sklar `sklar@student.net`
: Usando o MySQL a partir de PHP e Perl.

Alistair MacDonald `A.MacDonald@uel.ac.uk`
: Ainda não especificado, mas é flexível e pode lidar com Linux e, talvez, HP-UX. Tentará conseguir usuários para utilizar `mysqlbug`.

John Lyon `jlyon@imag.net`
: Questões sobre instalação do MySQL em sistemas Linux, usando ou arquivos '`.rpm`' ou compilando o fonter.

Lorvid Ltd. `lorvid@WOLFENET.com`
: Assuntos simples de contas/licença/suporte/direitos autorais

Patrick Sherrill `patrick@coconet.com`
: Questões sobre interfaces ODBC e VisualC++.

Randy Harmon `rjharmon@uptimecomputers.com`
: Questões sobre `DBD`, Linux, e algumas sintxe SQL.

C.3 Responsáveis pela Documentação e Tradução

As seguintes pessoas nos ajudaram com a escrita da documentação do MySQL e a tradução da documentação ou mensagens de erro no MySQL.

Paul DuBois
: Ajuda no progresso deste manual tornando-o correto e compreendivel. O que inclui rescrever o inglês do Monty e David em um inglês que todo mundo conhece.

Kim Aldale
: Ajudou a reescrever o inglês utilizado por Monty e Davis em inglês correto.

Michael J. Miller Jr. `mke@terrapin.turbolift.com`
: Pelo primeiro manual MySQL. E diversas grafia/linguagem corrigidas no FAQ (que virou o manual MySQL a muito tempo atras)

Yan Cailin
: Primeiro tradutor do Manual de Referência do MySQL em chinês simplificado no início de 2000, no qual a versão do código Big5 e HK (`http://mysql.hitstar.com/`) foram baseadas. Pagina pessoal em linuxdb.yeah.net (`http://linuxdb.yeah.net`).

Jay Flaherty `fty@mediapulse.com`
: Grande parte da seção Perl `DBI/DBD` no manual.

Paul Southworth `pauls@etext.org`, Ray Loyzaga `yar@cs.su.oz.au`
: Aprovação do Manual de Referência.

Therrien Gilbert `gilbert@ican.net`, Jean-Marc Pouyot `jmp@scalaire.fr`
: Mensagens de erro em Francês.

Petr Snajdr, `snajdr@pvt.net`
: Mensagens de erro em Tcheco.

Jaroslaw Lewandowski `jotel@itnet.com.pl`
: Mensagens de erro em Polonês

Miguel Angel Fernandez Roiz
: Mensagens de erro em Espanhol

Roy-Magne Mo `rmo@www.hivolda.no`
: Mensagens de erro em norueguês e teste da versão 3.21.#.

Timur I. Bakeyev `root@timur.tatarstan.ru`
: Mensagens de erro em russo.

`brenno@dewinter.com` & Filippo Grassilli `phil@hyppo.com`
: Mensagens de erro em italiano.

Dirk Munzinger `dirk@trinity.saar.de`
: Mensagens de erro em alemão.

Billik Stefan `billik@sun.uniag.sk`
: Mensagens de erro en eslovaco.

Stefan Saroiu `tzoompy@cs.washington.edu`
: Mensagens de erro em romeno.

Peter Feher
: Mensagens de erro em hungaro.

Roberto M. Serqueira
: Mensagens de erro em português.

Carsten H. Pedersen
: Mensgens de erro em dinamarquês.

Arjen G. Lentz
: Mensagens de erro em holandês, completando a tradução parcial mais cedo. (também trabalhou na consistencia e grafia).

C.4 Bibliotecas usadas e incluidas com o MySQL

A seguir está uma lista dos criadores da biblioteca que incluímos com o fonte do servidor MySQL para facilitar a compilação e instalação do MySQL. Somos muito agradecidos a todos os indivíduos que as criaram e têm feito a nossa vida mais fácil.

Fred Fish Pela sua excelente depuração de C e biblioteca trace. Monty fez pequenas melhoras nesta biblioteca (velocidade e opções adicionais).

Richard A. O'Keefe
Por sua biblioteca string de domínio público.

Henry Spencer
Pela sua biblioteca regex, usada em `WHERE column REGEXP regexp`.

Chris Provenzano
Pthreads portáveis no nível de usuário. Do direito de uso: Este produto inclui software desenvolvido por Chris Provenzano, pela Univesidade da Califórnia, Berkeley e colaboradores. Atualmente estamos usando a versão 1_60_beta6 corrigida pelo Monty (veja '`mit-pthreads/Changes-mysql`').

Jean-loup Gailly and Mark Adler
Pela biblioteca zlib (usada no MySQL para Windows).

Bjorn Benson
Por seu pacote safe_malloc (verificador de memória) que é usado quando você configura o MySQL com `--debug`.

Free Software Foundation
A biblioteca `readline` (para o cliente `mysql`).

The NetBSD fondation
O pacote `libedit` (usado opcionalmente pelo cliente de linha de comando `mysql`).

C.5 Pacotes que suportam o MySQL

A seguir encontra-se uma lista dos criadores/mantenedores de algumas das mais importantes APIs/pacotes/aplicações que muitas pessoas utilizam com o MySQL.

Não podemos listar todos os pacotes existentes aqui porque a lista seria muito difícil de manter. Para outros pacotes, vá ao portal do software em `http://www.mysql.com/portal/software`.

Tim Bunce, Alligator Descartes
Pela interface `DBD` (Perl).

Andreas Koenig `a.koenig@mind.de`
Pela interface Perl para o servidor MySQL.

Jochen Wiedmann `wiedmann@neckar-alb.de`
Por manter o módulo Perl `DBD::mysql`.

Eugene Chan `eugene@acenet.com.sg`
Por portar o PHP para o servidor MySQL.

Georg Richter
Teste do MySQL 4.1 e "caçador" de bugs. Nova extensão (API) `mysqli` do PHP 5.0 para uso com o MySQL 4.1 e acima.

Giovanni Maruzzelli `maruzz@matrice.it`
: Por portar iODBC (ODBC para Unix).

Xavier Leroy `Xavier.Leroy@inria.fr`
: O autor da LinuxThreads (usada pelo servidor MySQL no Linux).

C.6 Ferramentas que são usadas para criar o MySQL

Segue aqui uma lista de algumas das ferramentas que usamos para criar o MySQL. Nós a utilizamos para expressar nossos agradecimentos para aquelea que as criaram e sem as quais não poderiamos ter feito do MySQL o que ele é hoje.

Free Software Foundation
: De quem obtemos um excelente compilador (`gcc`), a biblioteca `libc` (de onde pegamos emprestado o '`strto.c`' para termos algum código funcionando em Linux)

Free Software Foundation
: From whom we got an excellent compiler (`gcc`), an excellent debugger (`gdb` and the `libc` library (from which we have borrowed '`strto.c`' to get some code working in Linux).

Free Software Foundation & The XEmacs development team
: For a really great editor/environment used by almost everybody at MySQL AB.

Julian Seward
: Author of `valgrind`, an excellent memory checker tool that has helped us find a lot of otherwise hard to find bugs in MySQL.

Dorothea Lütkehaus and Andreas Zeller
: For `DDD` (The Data Display Debugger) which is an excellent graphical frontend to `gdb`).

C.7 Responsáveis pelo Suporte do MySQL

Enquanto a `MySQL AB` montém todos os direitos autorais do `servidor MySQL` e do `manual MySQL`, desejamos apresentar as seguintes companias, que nos ajudaram financeiramente no desenvolvimento do `servidor MySQL`, nos pagando para desenvolver novos recursos ou nos dando hardware para o desenvolvimento do `servidor MySQL`.

VA Linux / Andover.net
: Replicações de fundos.

NuSphere Edição do manual MySQL.

Stork Design studio
: O site da MySQL usado entre 1998-2000.

Intel Contribuição para desenvolvimento nas plataformas Windows e Linux.

Compaq Contribuição no desenvolvimento do Linux/Alpha

SWSoft Desenvolvimento da versão embutida do `mysqld`.

FutureQuest
`--skip-show-database`

Apêndice D Histórico de Alterações do MySQL

Este apêndice lista as alterações de versão para versão no código fonte do MySQL.

Estamos agora trabalhando ativamente no MySQL 4.1 & 5.0 e só forneceremos correções de erros críticos para o MySQL 4.0 e MySQL 3.23. Atualizamos esta seção a medida que adicionamos novos recursos, para que assim todos possam acompanhar o desenvolvimento.

Nossa seção de TODO contém os planos adicionais que temos para as versões 4.1 e 5.0. Veja Seção 1.6 [TODO], Página 27.

Note que tendemos a atualizar o manual ao mesmo tempo em que fazemos as alterações no MySQL. Se você encontrar um versão listada aqui que você não pode encontrar na página de download do MySQL (`http://www.mysql.com/downloads/`), significa que a versão ainda não foi liberada!

A data mencionada com uma versão liberada é a data do último BitKeeper ChangeSet na qual esta distribuição particular foi baseada, e não a data em que os pacotes estavam disponíveis. Os binários estão disponíveis normalmente alguns dias após a data indicada do ChangeSet - contruir e testar todos os pacotes levam algum tempo.

D.1 Alterações na distribuição 5.0.0 (Development)

No momento, a versão 5.0 só está disponível em seu código fonte. Veja Seção 2.3.4 [Installing source tree], Página 100.

O seguinte log de alterações mostra o que já foi feito na árvore 5.0:

- Suporte básico a stored procedures (estilo SQL-99).
- Adicionado `SELECT INTO lista_de_vars`, que pode ser misturados, p.ex.: tipos locais e globais.
- O log de atualização está obsoleto (não é mais suportado). Ele está totlalmente substituído pelo log binário.
- Nomes de variáveis de usuários agora estão em caso insensitivo: se você fizer `SET @a=10;` então `SELECT @A;` retornará `10`. É claro que o conteúdo da variável ainda é caso sensitivo, apenas o seu nome é caso insensitivo.

D.2 Alterações na distribuição 4.1.x (Alpha)

A versão 4.1 do servidor MySQL inclui muitos melhoramentos e novos recursos. Os binários desta versão estão disponíveis para download em `http://www.mysql.com/downloads/mysql-4.1.html`.

- Subqueries:

```
SELECT * FROM t1 WHERE t1.a=(SELECT t2.b FROM t2);

SELECT * FROM t1 WHERE (1,2,3) IN (SELECT a,b,c FROM t2);
```

- Tabelas derivadas:

```
SELECT t1.a FROM t1, (SELECT * FROM t2) t3 WHERE t1.a=t3.a;
```

- Sintaxe `INSERT ... ON DUPLICATE KEY UPDATE ...`. Ela lhe permite fazer um `UPDATE` de um registro existente se a inserção criasse um valor duplicado em uma chave `PRIMARY` ou `UNIQUE`. (`REPLACE` lhe permite sobrescrever um registro existente, o que é totalmente diferente). Veja Seção 6.4.3 [INSERT], Página 579.
- Uma nova função de agrupamento `GROUP_CONCAT()`. Veja Seção 6.3.7 [Group by functions and modifiers], Página 555.
- Suporte a Unicode Extensivo (UTF8).
- Os conjuntos de caracteres podem ser definidos por colunas, tabelas e bancos de dados.
- Nova cache de chaves para tabelas MyISAM com vários parâmetros de ajustes. Você pode tem multiplas caches de cahves, índices precarregados em caches para batches ...
- Índices `BTREE` em tabelas `HEAP`.
- Suporte a OpenGIS (Dados Geográficos). Veja Cap“ptexi tulo 10 [Spatial extensions in MySQL], Página 733.
- `SHOW WARNINGS` exibe avisos para o último comando. Veja Seção 4.6.8.9 [`SHOW WARNINGS`], Página 323.
- Protocolo binário mais rápido com instruções prepardas e ligação de parâmetros. Veja Seção 12.1.4 [C API Prepared statements], Página 828.
- Agora você pode executar várias instruções com uma única chamada a API C e de uma vez e então ler o resultado Veja Seção 12.1.8 [C API multiple queries], Página 855.
- Create Table: `CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tabela LIKE tabela`.
- Comando `HELP` baseado no servidor que pode ser usado no cliente `mysql` de linha de comando (e outros clientes) para obter ajuda para comandos SQL.

Para uma lista completa das atualizações, veja a seção de alterações para cada distribuição 4.1.x individual.

D.2.1 Alterações na distribuição 4.1.2 (not released yet)

Functionality added or changed:

- `ENGINE` is now a synonym for the `TYPE` option for `CREATE TABLE` and `ALTER TABLE`.
- Added `init_connect` and `init_slave` server variables. The values should be SQL statements to be executed when each client connects or each time a slave's SQL thread starts, respectively.
- C API enhancement: `SERVER_QUERY_NO_INDEX_USED` and `SERVER_QUERY_NO_GOOD_INDEX_USED` flags are now set in `server_status` field of `MYSQL` structure. It is these flags that make the query to be logged as slow if '`mysqld`' was started with `--log-slow-queries --log-queries-not-using-indexes`.

Bugs fixed:

- Fixed a bug with the `INTERVAL()` function when 8 or more comparison arguments are provided. (Bug #1561)
- Packaging: Fixed a bug in the Mac OS PKG `postinstall` script (`mysql_install_db` was called with an obsolete argument).

- Packaging: Added missing file '`mysql_create_system_tables`' to the server RPM package. This bug was fixed for the 4.1.1 RPMs by updating the MySQL-server RPM from `MySQL-server-4.1.1-0` to `MySQL-server-4.1.1-1`. The other RPMs were not affected by this change.
- Fixed a bug in '`myisamchk`' and `CHECK TABLE` that sometimes resulted in a spurious error `Found key at page ..... that points to record outside datafile` for a table with a `FULLTEXT` index. (Bug #1977)
- Fixed a hang in full-text indexing of strings in multi-byte (all besides `utf8`) charsets. (Bug #2065)
- Fixed a crash in full-text indexing of UTF-8 data. (Bug #2033)
- Replication: a rare race condition in the slave SQL thread that could lead to an incorrect complaint that the relay log is corrupted. (Bug #2011)
- Replication: if an administrative command on a table (`OPTIMIZE TABLE`, `REPAIR TABLE` etc) was run on the slave, this could sometimes stop the slave SQL thread (this did not lead to any corruption; one just had to type `START SLAVE` to get replication going again). (Bug #1858)
- Replication: in the slave SQL thread, a multi-table `UPDATE` could produce an incorrect complaint that some record was not found in one table, if the `UPDATE` was preceded by a `INSERT ... SELECT`. (Bug #1701)

D.2.2 Alterações na distribuição 4.1.1 (01 de Dez de 2003)

Funcionalidades adicionadas ou alteradas:

- Added `IGNORE` option for `DELETE` statement.
- The MySQL source distribution now also includes the MySQL Internals Manual '`internals.texi`'.
- Added `mysql_set_server_option()` C API client function to allow multiple statement handling in the server to be enabled or disabled.
- The `mysql_next_result()` C API function now returns `-1` if there are no more result sets.
- Renamed `CLIENT_MULTI_QUERIES` connect option flag to `CLIENT_MULTI_STATEMENTS`. To allow for a transition period, the old option will continue to be recognized for a while.
- Require `DEFAULT` before table and database default character set. This enables us to use `ALTER TABLE table_name ... CHARACTER SET=...` to change the character set for all `CHAR`, `VARCHAR`, and `TEXT` columns in a table.
- Added `MATCH ... AGAINST( ... WITH QUERY EXPANSION)` and the `ft_query_expansion_limit` server variable.
- Removed unused `ft_max_word_len_for_sort` server variable.
- Full-text search now supports multi-byte character sets and the Unicode `utf8` character set. (The Unicode `ucs2` character set is not yet supported.)
- Phrase search in `MATCH ... AGAINST ( ... IN BOOLEAN MODE)` no longer matches partial words.

- Added aggregate function `BIT_XOR()` for bitwise XOR operations.
- Replication over SSL now works.
- The `START SLAVE` statement now supports an `UNTIL` clause for specifying that the slave SQL thread should be started but run only until it reaches a given position in the master's binary logs or in the slave's relay logs.
- Produce warnings even for single-row `INSERT` statements, not just for multiple-row `INSERT` statements. Previously, it was necessary to set `SQL_WARNINGS=1` to generate warnings for single-row statements.
- Added `delimiter` (`\d`) command to the `mysql` command-line client for changing the statement delimiter (terminator). The default delimiter is semicolon.
- `CHAR`, `VARCHAR`, and `TEXT` columns now have lengths measured in characters rather than in bytes. The character size depends on the column's character set. This means, for example, that a `CHAR(n)` column for a multi-byte character set will take more storage than before. Similarly, index values on such columns are measured in characters, not bytes.
- `LIMIT` no longer accepts negative arguments (they used to be treated as very big positive numbers before).
- The `DATABASE()` function now returns `NULL` rather than the empty string if there is no database selected.
- Added `--sql-mode=NO_AUTO_VALUE_ON_ZERO` option to suppress the usual behaviour of generating the next sequence number when zero is stored in an `AUTO_INCREMENT` column. With this mode enabled, zero is stored as zero; only storing `NULL` generates a sequence number.
- **Warning: Incompatible change!** Client authentication now is based on 41-byte passwords in the `user` table, not 45-byte passwords as in 4.1.0. Any 45-byte passwords created for 4.1.0 must be reset after running the `mysql_fix_privilege_tables` script.
- Added MySQL Server option and global variable 'secure-auth' that disallows authentication for accounts that have old (pre-4.1.1) passwords.
- Added MySQL command line client option 'secure-auth'. If this option is set, client will refuse to send password in old (pre-4.1.1) format.
- **Warning: Incompatible change!** Renamed the C API `mysql_prepare_result()` function to `mysql_get_metadata()` as the old name was confusing.
- Added `DROP USER 'username'@'hostname'` statement to drop an account that has no privileges.
- The interface to aggregated UDF functions has changed a bit. You must now declare a `xxx_clear()` function for each aggregate function `XXX()`.
- The `CONCAT_WS()` function no longer skips empty strings.
- Added new `ADDTIME()`, `DATE()`, `DATEDIFF()`, `LAST_DAY()`, `MAKEDATE()`, `MAKETIME()`, `MICROSECOND()`, `SUBTIME()`, `TIME()`, `TIMEDIFF()`, `TIMESTAMP()`, `UTC_DATE()`, `UTC_TIME()`, `UTC_TIMESTAMP()`, and `WEEKOFYEAR()` functions.
- Added new syntax for `ADDDATE()` and `SUBDATE()`. The second argument now may be a number representing the number of days to be added to or subtracted from the first date argument.

- Added new `type` values `DAY_MICROSECOND`, `HOUR_MICROSECOND`, `MINUTE_MICROSECOND`, `SECOND_MICROSECOND`, and `MICROSECOND` for `DATE_ADD()`, `DATE_SUB()`, and `EXTRACT()`.
- Added new `%f` microseconds format specifier for `DATE_FORMAT()` and `TIME_FORMAT()`.
- All queries in which at least one `SELECT` does not use indexes properly now are written to the slow query log when long log format is used.
- It is now possible to create a `MERGE` table from `MyISAM` tables in different databases. Formerly, all the `MyISAM` tables had to be in the same database, and the `MERGE` table had to be created in that database as well.
- Adicionada as novas funções `COMPRESS()`, `UNCOMPRESS()` e `UNCOMPRESSED_LENGTH()`.
- Ao fazer `SQL SQL_MODE=#`, para um modo complexo (como `ANSI`) agora atualizamos a variável `SQL_MODE` para incluir todas as opções que o modo exige.
- Adicionada a função `ROLLUP` OLAP (Online Analytical Processing - Processamento Analítico Online), que lhe dá um resumo para cada nível `GROUP BY`.
- Adicionado os códigos `SQLSTATE` para todos os erros do servidor.
- Adicionado `mysql_sqlstate()` e `mysql_stmt_sqlstate()` que retornam o código de erro `SQLSTATE` para o último erro.
- `--lower-case-table-names=1` agora também faz a aliases caso insensitivo. (Bug #534)
- Colunas `TIME` com valor de horas maior do que 24 eram retornadas incorretamente para o cliente.
- As instruções `ANALYZE`, `OPTIMIZE`, `REPAIR` e `FLUSH` são agora armazenados no log binário e assim replicados para o slave. Este registro não ocorre se a palavra chave opcional `NO_WRITE_TO_BINLOG` (ou seu alias `LOCAL`) for usada. As exceções são que `FLUSH LOGS`, `FLUSH MASTER`, `FLUSH SLAVE` e `FLUSH TABLES WITH READ LOCK`, não são registrados no log em qualquer caso. Para uma sintaxe completa, veja Seção 4.6.4 [`FLUSH`], Página 299.
- Nova variável global `RELAY_LOG_PURGE` para habilitar ou desabilitar automaticamente a remoção de relay logs.
- `LOAD DATA` agora produz avisos que podem ser buscados com `SHOW WARNINGS`.
- Adicionado o suporte a sintaxe `CREATE TABLE nome_tabela (LIKE nome_tabela2)`.
- `CREATE TABLE nome_tabela (...) TYPE=storage_engine` agora gera um aviso se o mecanismo de armazenamento não for respeitado. A tabela ainda é criada como `MyISAM`, como antes.
- Muitas sub selectas são muito mais rápidas que antes.
- Disabled the `PURGE LOGS` statement that was added in in version 4.1.0. The statement now should be issued as `PURGE MASTER LOGS` or `PURGE BINARY LOGS`.
- Added `SHOW BDB LOGS` as an alias for `SHOW LOGS`.
- Added `SHOW MASTER LOGS` (which had been deleted in version 4.1.0) as an alias for `SHOW BINARY LOGS`.
- Added `Slave_IO_State` and `Seconds_Behind_Master` columns to the output of `SHOW SLAVE STATUS`. `Slave_IO_State` indicates the state of the slave I/O thread, and

`Seconds_Behind_Master` indicates the number of seconds by which the slave is late compared to the master.

- `--lower-case-table-names=1` now also makes aliases case insensitive. (Bug #534)

Bugs corrigidos:

- Fixed merging types and length of fields in `UNION`
- Fixed a bug in privilege handling that caused connections from certain IP addresses to be assigned incorrect database-level privileges. A connection could be assigned the database privileges of the previous successful authentication from one of those IP addresses, even if the IP address username and database name were different. (Bug #1636)
- Error-handling functions were not called properly when an error resulted from `[CREATE | REPLACE| INSERT] ... SELECT` statements.
- `HASH`, `BTREE`, `RTREE`, `ERRORS`, and `WARNINGS` no longer are reserved words. (Bug #724)
- Fix for bug in `ROLLUP` when all tables were `const` tables. (Bug #714)
- Fixed a bug in `UNION` that prohibited `NULL` values from being inserted into result set columns where the first `SELECT` of the `UNION` retrieved `NOT NULL` columns.
- Fixed name resolution of columns of reduced subqueries in unions. (Bug #745)
- Fixed memory overrun in subqueries in select list with `WHERE` clause bigger than outer query `WHERE` clause. (Bug #726)
- Fixed a bug that caused `MyISAM` tables with `FULLTEXT` indexes created in 4.0.x to be unreadable in 4.1.x.
- Fixed a data loss bug in `REPAIR TABLE ... USE_FRM` when used with tables that contained `TIMESTAMP` columns and were created in 4.0.x.
- Fixed reduced subquery processing in `ORDER BY/GROUP BY` clauses. (Bug #442)
- Fixed name resolution of outer columns of subquery in `INSERT/REPLACE` statements. (Bug #446)
- Fixed bug in marking columns of reduced subqueries. (Bug #679)
- Fixed a bug that made `CREATE FULLTEXT INDEX` syntax illegal.
- Fixed a crash when a `SELECT` that required a temporary table (marked by `Using temporary` in `EXPLAIN` output) was used as a derived table in `EXPLAIN` command. (Bug #251)
- Fixed a rare table corruption bug in `DELETE` from a big table with a **new** (created by MySQL-4.1) fulltext index.
- `LAST_INSERT_ID()` now returns 0 if the last `INSERT` statement didn't insert any rows.
- Corrigido a perda dos últimos caracteres na saída da função (bug #447)
- Corrigido um erro de replicação raro quando um transação extendia em dois ou mais relay logs e o escravo era parada enquanto ele estava executando a parte da transação que estava no segundo relay log ou em um adicional. Então a replicação parava no inicio do segundo relay log ou adicional, o que estava incorreto. (ele deve parar no `BEGIN`, no primeiro relay log). (Bug #53)
- Agora `CONNECTION_ID()` é replicado apropriadamente (bug #177).
- A nova função `PASSWORD()` na versão 4.1 é replicada apropriadamente (bug #344).

- Corrigida a dupla liberação de memória
- Corrigido um erro em `UNION` envolvendo tabelas temporárias.
- Corrigido um erro de falha em `DERIVED TABLES` quando `EXPLAIN` é usado em um `DERIVED TABLES` com um join
- Corrigido um erro de falha no `DELETE` com `ORDER BY` e `LIMIT` causado pala inicialização do vetor do ponteiro de referências.
- Corrigido um erro na função `USER()` causado pelo erro no tamanho da string alocada
- Corrigido um erro de falha quando se tentava criar uma tabela com coluna do tipo `GEOMETRY` com um mecanismo de armazenamenti que não a suporta.
- Corrigido um erro de falha no `UNION` causado pela lista de select vazia e um campo não existente sendo usado em algumas das instruções `SELECTs` individuais.
- Corrigido um erro de replicação com um master na versão 3.23 e um slave na 4.0: o slave perdia a replicação de tabelas temporárias se `FLUSH LOGS` era executado no master (Bug #254).
- Corrigido um bug de segurança: Um servidor compilado ser suporte a SSL ainda permitia conexões de usuários que possuiam a opção `REQUIRE SSL` especificado para as suas contas.
- Quando um usuário indefinido era usado em uma atualização de consulta no master (como `INSERT INTO t VALUES(@a)` onde `@a` nunca havia sido definido por esta conexão), então o slave podia replicar a consulta de forma incorreta se uma transação anterior no master usava uma variável de usuário de mesmo nome. (Bug #1331)
- Corrigido um erro com instruções preparadas: O uso do parâmetro `?` de instruções preparadas como argumento de certas funções e cláusulas fazia com que o servidor falhasse durante chamadas `mysql_prepare()`. (Bug #1500)
- Corrigido um erro com instruções preparadas: depois da chamada de mysql_stmt_prepare, colchetes são permitidos em todas as instruções consequentes, mesmo se eles não forem preparados (bug #1946)

D.2.3 Alterações na distribuição 4.1.0 (03 Apr 2003: Alpha)

Funcionalidades adicionadas ou alteradas:

- Nova autenticação do cliente, mais segura, baseada em senha de 45-byte na tabela `user`.
- Nova função `CRC32()` para calcular valor de verificação de redundância cíclica.
- No Windows, agora estamos usando memória compartilhada para comunicar entre servidor e cliente quando eles estão executando na mesma máquina e você está conectando a `localhost`.
- `REPAIR` das tabelas MyISAM agora usam menos espaço temporário em disco ao ordenar as colunas de caracteres.
- A verificação de `DATE/DATETIME` agora é um bit estritamente para suportar a habilidade de deitiguir automaticamente entre date, datetime e time com microsegundos. Por exemplo, tipos de dados `YYYYMMDD HHMMDD` não são mais suportados; deve-se também ter separadores entre as partes `DATE/TIME` ou não.

- Ajuda do lado do servidor para todas as funções do MySQL. Pode-se agora digitar `help week` no cliente `mysql` e conseguir ajuda para a função `week()`.
- Adionada a nova função da API C `mysql_get_server_version()`.
- Corrigido um buh na `libmysqlclient` que buscava campos padrões.
- Corrigido um bug no cliente '`mysql.cc`' ao ignorar comentários
- Adicionado o método `record_in_range()` para tabelas `MERGE` poderem escolher o índice certo quando houverem muitos para serem escolhidos.
- A replicação agora funciona com `RAND()` e variáveis de usuários `@var`.
- Permite-se alterar o modo para `ANSI_QUOTES` com o servidor no ar.
- Agora pode se matar `EXPLAIN SELECT`. Veja Seção 4.6.7 [`KILL`], Página 301.
- Agora pode se matar `REPAIR TABLE`. Veja Seção 4.6.7 [`KILL`], Página 301.
- Permiti-se especificar lista de chaves vazias para `USE INDEX`, `IGNORE INDEX` e `FORCE INDEX`.
- Agora `DROP TEMPORARY TABLE` apenas apaga tabelas temporárias e não finaliza transações.
- Adicionado suporte para `UNION` em tabelas derivadas.
- **Warning: Alteração imcompatível!** `TIMESTAMP` agora é retornado comi uma string do tipo `'YYYY-MM-DD HH:MM:SS'` e tamanhos de timestamp diferentes não são suportados.

  Esta alteração era necessária para compatibilidade com o padrão SQL. Em uma versão futura, uma alteração adicional será feita (compatível co esta alteração), permitindo que o tamanho do timestamp indique o número de dígitos desejado para a fração de segundos.
- Novo protocolo cliente/servidor mais rápido que suporta instruções preparadas, limitar parâmetros e colunas de resultados, transferância binaria de dados, avisos.
- Adicionado nome de banco de dados e de nomes reais de tabela (no caso de alias) à estrutura `MYSQL_FIELD`.
- Consultas multi linhas: Agora você pode executar diversas consultas de uma vez e então ler o resultados.
- Em `CREATE TABLE foo (a INT not null primary key)` a palavra `PRIMARY` agora é opcional.
- Em `CREATE TABLE` o atributo `SERIAL` agora é um alias para `BIGINT NOT NULL AUTO_INCREMENT UNIQUE`.
- `SELECT ... FROM DUAL` é um alias para `SELECT ...`. (Para ser compatível com alguns outros bancos de dados).
- Se é criado um `CHAR/VARCHAR` muito grande, ele á alterado automaticamente para `TEXT` ou `BLOB`; Será exibido um aviso neste caso.
- POde-se especificar os tipos `BLOB/TEXT` diferentes com a sintaxe `BLOB(tamanho)` e `TEXT(tamanho)`. O MySQL irá alterá-los automaticamente para um dos tipos internos `BLOB/TEXT`.
- `CHAR BYTE` é um alias para `CHAR BINARY`.
- `VARCHARACTER` é um alias para `VARCHAR`.
- Novos operadores `inteiro MOD inteiro` e `inteiro DIV inteiro`.

- Adicionado `SERIAL DEFAULT VALUE` como um alias para `AUTO_INCREMENT`.
- Adicionado `TRUE` e `FALSE` como alias para 1 e 0, respectivamente.
- Agora aliases são forçados em tabelas dferivadas, como no SQL-99.
- orrigido `SELECT .. LIMIT 0` para retornar a contagem aproriada de linhas para `SQL_CALC_FOUND_ROWS`.
- Pode-se especificar muitos diretórios temporários para serem usados de modo round-robin com: `--tmpdir=nomedir1:nomedir2:nomedir3`.
- Subqueries: `SELECT * from t1 where t1.a=(SELECT t2.b FROM t2)`.
- Tabelas derivadas:

```
SELECT a.col1, b.col2
       FROM (SELECT MAX(col1) AS col1 FROM root_table) a,
       other_table b
       WHERE a.col1=b.col1;
```

- Conjuntos de caracteres a serem definidos por colunas, tabelas e banco de dados.
- Suporte a Unicode (UTF8).
- Nova sintaxe `CONVERT(... USING ...)` para conversão de valores strings entre conjunto de caracteres.
- Índices `BTREE` em tabelas `HEAP`.
- Servidor embutido mais rápido (novo protocolo de comunicação interno).
- Pode-se adicionar um comentário por coluna em `CREATE TABLE`.
- `SHOW FULL COLUMNS FROM nome_tabela` exibe os comentários das colunas.
- `ALTER DATABASE`.
- Suporte a GIS (dados geometricos). Veja Cap“ptexi tulo 10 [Spatial extensions in MySQL], Página 733.
- `SHOW [COUNT(*)] WARNINGS` exibe avisos sobre o último comnado.
- Pode se especificar um tipo de coluna para em um `CREATE TABLE ... SELECT` definindo a coluna na parte `CREATE`.

```
CREATE TABLE foo (um tinyint não nulo) SELECT b+1 AS 'a' FROM bar;
```

- `expr SOUNDS LIKE expr` é o mesmo que `SOUNDEX(expr)=SOUNDEX(expr)`.
- Adicionada nova função `VARIANCE(expr)` que retorna a variância de `expr`
- Pode se criar um tabela a partir de uma existente usando `CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tabela (LIKE tabela)`. A tabela também pode ser normal ou temporária.
- Novas opções `--reconnect` e `--disable-reconnect` para o cliente `mysql`, para reconectar automaticamente ou não se a conexão for perdida.
- `START SLAVE` (`STOP SLAVE`) não retorna mais um erro se o slave já está iniciado (parado); ele retorns um aviso.
- `SLAVE START` e `SLAVE STOP` não é mais aceitada pelo analisador de consulta; use `START SLAVE` e `STOP SLAVE` em seu lugar.

D.3 Alterações na distribuição 4.0.x (Production)

A versão 4.0 do servidor MySQL inclui muitos aprimoramentos e novos recursos:

- O tipo de tabela `InnoDB` agora está incluído no binário padrão, adicionando transações, lock de linha e chaves estrangeiras. Veja Seção 7.5 [`InnoDB`], Página 643.
- Uma cache de consultas, oferecendo um grande aumento da performance para muitas aplicações. Armazenando resultados completos, mais tarde consultas idênticas podem ser retornadas instataneamente. Veja Seção 6.9 [Query Cache], Página 625.
- Melhora na indexação full-text com modo booleano, truncamento e busca de frase. Veja Seção 6.8 [Fulltext Search], Página 619.
- Melhor das tabelas `MERGE`, suportando `INSERT`s e `AUTO_INCREMENT`. Veja Seção 7.2 [`MERGE`], Página 638.
- Sintaxe `UNION` em `SELECT`. Veja Seção 6.4.1.2 [`UNION`], Página 569.
- Instruções `DELETE` multi-tabelas. Veja Seção 6.4.5 [`DELETE`], Página 585.
- `libmysqld`, a biblioteca do servidor embutido. Veja Seção 12.1.15 [`libmysqld`], Página 864.
- Opções adicionais para o privilégio `GRANT` para maior controle e segurança. Veja Seção 4.4.1 [`GRANT`], Página 254.
- Gerenciamento dos recursos dos usuários no sistema `GRANT`, particularmente útil para provedores e outro fornecedores de hospedagem. Veja Seção 4.4.7 [User resources], Página 265.
- Variáveis de servidores dinâmicas, permitindo que alterações na configuração sejam feitas ser precisar derrubar o servidor. Veja Seção 5.5.6 [`SET OPTION`], Página 460.
- Melhora do código da replicação e seus recursos. Veja Seção 4.11 [Replication], Página 378.
- Novas funções e opções numerosas.
- Alterações do código existente para melhora da performance e confiabilidade.

Para uma lista completa de alterações, visite a seção para cada distribuição 4.0.x individual.

D.3.1 Alterações na distribuição 4.0.17 (not released yet)

Functionality added or changed:

- Allow spaces in windows service names.
- Changed the default Windows service name for `mysqld` from `MySql` to `MySQL`. This should not affect usage, because service names are not case sensitive.
- When you install `mysqld` as a service on Windows systems, `mysqld` will read startup options in option files from the option group with the same name as the service name. (Except when the service name is `MySQL`).

Bugs fixed:

- Fixed bug #1335 when filesort was never shown in EXPLAIN if query contained ORDER BY NULL clause.
- Fixed invalidation of whole query cache on `DROP DATABASE`. (Bug #1898)

- Fixed bug in range optimizer that caused wrong results for some not likely `AND/OR` queries. (Bug #1828)
- Fixed a crash in `ORDER BY` when ordering by expression and identifier. (Bug #1945)
- Fixed a crash in an open `HANDLER` when an `ALTER TABLE` was executed in a different connection. (Bug #1826)
- Fixed a bug in `trunc*` operator of full-text search which sometimes caused MySQL not to find all matched rows.
- Fixed bug in zero prepending to `DECIMAL` column type.
- Fixed optimiser bug, introduced in 4.0.16, when `REF` access plan was preferred to more efficient `RANGE` on another column.
- Fixed problem when installing a MySQL server as a Windows service using a command of the form `mysqld --install mysql --defaults-file=path-to-file`.
- Fixed an incorrect result from a query that uses only `const` tables (such as one-row tables) and non-constant expression (such as `RAND()`). (Bug #1271)
- Fixed bug when the optimiser did not take `SQL_CALC_FOUND_ROWS` into account if `LIMIT` clause was present. (Bug #1274)
- `mysqlbinlog` now asks for a password at the console when the `-p` or `--password` option is used with no argument. This is consistent with the way that other clients such `mysqladmin` and `mysqldump` already behave. **Note:** A consequence of this change is that it is no longer possible to invoke `mysqlbinlog` as `mysqlbinlog -p pass_val` (with a space between the `-p` option and the following password value). (Bug #1595)
- Bug accidentally introduced in 4.0.16 where the slave SQL thread deleted its replicated temporary tables when `STOP SLAVE` was issued.
- In a "chain" replication setup `A->B->C`, if 2 sessions on A updated temporary tables of the same name at the same time, the binary log of `B` became incorrect, resulting in `C` becoming confused. (Bug #1686)
- In a "chain" replication setup `A->B->C`, if `STOP SLAVE` was issued on `B` while it was replicating a temporary table from `A`, then when `START SLAVE` was issued on `B`, the binary log of `B` became incorrect, resulting in `C` becoming confused. (Bug #1240)
- When `MASTER_LOG_FILE` and `MASTER_LOG_POS` were not specified, `CHANGE MASTER` used the coordinates of the slave I/O thread to set up replication, which broke replication if the slave SQL thread lagged behind the slave I/O thread. This caused the slave SQL thread to lose some events. The new behaviour is to use the coordinates of the slave SQL thread instead. Veja Seção 4.11.8.1 [`CHANGE MASTER TO`], Página 402. (Bug #1870)
- Now if integer is stored or converted to `TIMESTAMP` or `DATETIME` value checks of year, month, day, hour, minute and second ranges are performed and numbers representing illegal timestamps are converted to 0 value. This behaviour is consistent with manual and with behaviour of string to `TIMESTAMP/DATETIME` conversion. (Bug #1448)
- Fixed bug when `BIT_AND()` and `BIT_OR()` group functions returned incorrect value if `SELECT` used a temporary table and no rows were found. (Bug #1790).
- `BIT_AND()` is now unsigned in all contexts. This means that it will now return 18446744073709551615 (= 0xffffffffffffffff) instead of -1 if there were no rows in the result.

- Fixed bug with `BIT_AND()` still returning signed value for an empty set in some cases. (Bug #1972)
- Fixed bug with `^` (XOR) and `>>` (bit shift) still returning signed value in some cases. (Bug #1993)
- Replication: a rare race condition in the slave SQL thread, which could lead to a wrong complain that the relay log is corrupted. (Bug #2011)
- Replication: if an administrative command on a table (`OPTIMIZE TABLE`, `REPAIR TABLE` etc) was run on the slave, this could sometimes stop the slave SQL thread (this did not led to any corruption; one just had to type `START SLAVE` to get replication going again). (Bug #1858)
- Replication: in the slave SQL thread, a multi-table `UPDATE` could produce a wrong complain that some record was not found in one table, if the `UPDATE` was preceded by a `INSERT ... SELECT`. (Bug #1701)

D.3.2 Alterações na distribuição 4.0.16 (17 Out 2003)

Funcionalidades adicionadas ou alteradas:

- Write memory allocation information to error log when doing `mysqladmin debug`. This only works on system that support the `mallinfo()` call (like newer Linux systems).
- Added the following new server variables to allow more precise memory allocation: `range_alloc_block_size`, `query_alloc_block_size`, `query_prealloc_size`, `transaction_alloc_block_size`, and `transaction_prealloc_size`.
- `mysqlbinlog` now reads option files. To make this work one must now specify `--read-from-remote-server` when reading binary logs from a MySQL server. (Note that using a remote server is deprecated and may disappear in future `mysqlbinlog` versions).
- Block `SIGPIPE` signals also for non-threaded programs. The blocking is moved from `mysql_init()` to `mysql_server_init()`, which is automatically called on the first call to `mysql_init()`.
- Added `--libs_r` and `--include` options to `mysql_config`.
- New `'>` prompt for `mysql`. This prompt is similar to the `'>` and `">` prompts, but indicates that an identifier quoted with backticks was begun on an earlier line and the closing backtick has not yet been seen.
- Atualizado o `mysql_install_db` para poder usar o endereço de IP da máquina local em vez do nome da máquina ao criar as tabelas de permissões iniciais de `skip-name-resolve` foi especificado. Esta opção pode ser útil no FreeBSD para evitar problemas de segurança de threads com o resolver de bibliotecas do FreeBSD. (Obrigado a Jeremy Zawodny pelo patch)
- A documentation change: Added a note that when backing up a slave, it is necessary also to back up the '`master.info`' and '`relay-log.info`' files, as well as any '`SQL_LOAD-*`' files located in the directory specified by the `--slave-load-tmpdir` option. All these files are needed when the slave resumes replication after you restore the slave's data.

Bugs corrigidos:

- Fixed a spurious error `ERROR 14: Can't change size of file (Errcode: 2)` on Windows in `DELETE FROM table_name` without a `WHERE` clause or `TRUNCATE TABLE table_name`, when `table_name` is a `MyISAM` table. (Bug #1397)
- Fixed a bug that resulted in `thr_alarm queue is full` warnings after increasing the `max_connections` variable with `SET GLOBAL`. (Bug #1435)
- Made `LOCK TABLES` to work when `Lock_tables_priv` is granted on the database level and `Select_priv` is granted on the table level.
- Fixed crash of `FLUSH QUERY CACHE` on queries that use same table several times (Bug #988).
- Fixed core dump bug when setting an enum system variable (such as `SQL_WARNINGS`) to `NULL`.
- Extended the default timeout value for Windows clients from 30 seconds to 1 year. (The timeout that was added in MySQL 4.0.15 was way too short). This fixes a bug that caused `ERROR 2013: Lost connection to MySQL server during query` for queries that lasted longer than 30 seconds, if the client didn't specify a limit with `mysql_options()`. Users of 4.0.15 on Windows should upgrade to avoid this problem.
- More "out of memory" checking in range optimiser.
- Fixed and documented a problem when setting and using a user variable within the same `SELECT` statement. (Bug #1194).
- Fixed bug in overrun check for `BLOB` values with compressed tables. This was a bug introduced in 4.0.14. It caused MySQL to regard some correct tables containing `BLOB` values as corrupted. (Bug #770, Bug #1304, and maybe Bug #1295)
- `SHOW GRANTS` showed `USAGE` instead of the real column-level privileges when no table-level privileges were given.
- When copying a database from the master, `LOAD DATA FROM MASTER` dropped the corresponding database on the slave, thus erroneously dropping tables that had no counterpart on the master and tables that may have been excluded from replication using `replicate-*-table` rules. Now `LOAD DATA FROM MASTER` no longer drops the database. Instead, it drops only the tables that have a counterpart on the master and that match the `replicate-*-table` rules. `replicate-*-db` rules can still be used to include or exclude a database as a whole from `LOAD DATA FROM MASTER`. A database will also be included or excluded as a whole if there are some rules like `replicate-wild-do-table=db1.%` or `replicate-wild-ignore-table=db1.%`, as is already the case for `CREATE DATABASE` and `DROP DATABASE` in replication. (Bug #1248)
- Fixed a bug where `mysqlbinlog` crashed with a segmentation fault when used with the `-h` or `--host` option. (Bug #1258)
- Fixed a bug where `mysqlbinlog` crashed with a segmentation fault when used on a binary log containing only final events for `LOAD DATA`. (Bug #1340)
- Fixed compilation problem when compiling with OpenSSL 0.9.7 with disabled old DES support (If `OPENSSL_DISABLE_OLD_DES_SUPPORT` option was enabled).
- Fixed a bug when two (or more) MySQL servers were running on the same machine, and they were both slaves, and at least one of them was replicating some `LOAD DATA INFILE` command from its master. The bug was that one slave MySQL server sometimes deleted the '`SQL_LOAD-*`' files (used for replication of `LOAD DATA INFILE` and located

in the `slave-load-tmpdir` directory, which defaults to `tmpdir`) belonging to the other slave MySQL server of this machine, if these slaves had the same `slave-load-tmpdir` directory. When that happened, the other slave could not replicate `LOAD DATA INFILE` and complained about not being able to open some `SQL_LOAD-*` file. (Bug #1357)

- If `LOAD DATA INFILE` failed for a small file, the master forgot to write a marker (a `Delete_file` event) in its binary log, so the slave could not delete 2 files ('`SQL_LOAD-*.info`' and '`SQL_LOAD-*.data`' from its `tmpdir`. (Bug #1391)
- On Windows, the slave forgot to delete a `SQL_LOAD-*.info` file from `tmpdir` after successfully replicating a `LOAD DATA INFILE` command. (Bug #1392)
- When a connection terminates, MySQL writes `DROP TEMPORARY TABLE` statements to the binary log for all temporary tables which the connection had not explicitely dropped. MySQL forgot to backquote the database and table names in the statement. (Bug #1345)
- On some 64-bit machines (some HP-UX and Solaris machines), a slave installed with the 64-bit MySQL binary could not connect to its master (it connected to itself instead). (Bug #1256, Bug #1381)
- Code was introduced in MySQL 4.0.15 for the slave to detect that the master had died while writing a transaction to its binary log. This code reported an error in a legal situation: When the slave I/O thread was stopped while copying a transaction to the relay log, the slave SQL thread would later pretend that it found an unfinished transaction. (Bug #1475)

D.3.3 Alterações na distribuição 4.0.15 (03 Sep 2003)

IMPORTANT:

If you are using this release on Windows, you should upgrade at least your clients (any program that uses `libmysql.lib`) to 4.0.16 or above. This is because the 4.0.15 release had a bug in the Windows client library that causes Windows clients using the library to die with a `Lost connection to MySQL server during query` error for queries that take more than 30 seconds. This problem is specific to Windows; clients on other platforms are unaffected.

Funcionalidades adicionadas ou alteradas:

- O `mysqldump` agora coloca todos os identificadores corretamente entre aspas ao conectar com o servidor. Isto assegura que durante o processo de dump, O `mysqldump` nunca enviará consultas ao servidor que resultam em um erro de sintaxe. Este problema **não** está relacionado a saída do programa `mysqldump`, que não foi alterado. (Bug #1148)
- Altera a informação de metadados do resultado e assim `MIN()` e `MAX()` informamm que eles podem retornar `NULL` (isto é verdade porque um conjunto vazio retornará `NULL`). (Bug #324)
- Produz uma mensagem de erro no Windows se um segundo servidor `mysqld` é iniciado na mesma porta TCP/IP que um servidor `mysqld` já em execução.
- As variveis do servidor `mysqld wait_timeout`, `net_read_timeout` e `net_write_timeout` agora funcionam no Windows. Agora pode-se também definir o tempo limite de leitura e escrita em clientes Windows com a opção `mysql_options()`

- Adicionada a opção `--sql-mode=NO_DIR_IN_CREATE` para tornar possível para os slaves ignorarem as opções `INDEX DIRECTORY` e `DATA DIRECTORY` dadas para `CREATE TABLE`. Quando ele está ligado, `SHOW CREATE TABLE` não exibirá os diretórios dados.
- `SHOW CREATE TABLE` agora exibe as opções `INDEX DIRECTORY` e `DATA DIRECTORY`, se eles fossem especificados quando a tabela era criada.
- A variável do servidor `open_files_limit` agora exibe o limite de arquivos abertos real.
- `MATCH ... AGAINST()` em modo de linguagem natural agora tratam de palavra presentes em mais de 2,000,000 linhas como stopwords.
- As imagens do disco de instalação do Mac OS X agora incluem um pacote '`MySQLStartupItem.pkg`' adicional que habilita a inicialização automática do MySQL no boot do sistema. Veja Seção 2.1.3 [Mac OS X installation], Página 71.
- A maioria da documentação incluída na distribuição tar do binário (`.tar.gz`) foi movida para o subdiretório `docs`. Veja Seção 2.2.5 [Installation layouts], Página 83.
- O manual agora está incluído com um arquivo `info` tradicional na distribuição binária. (Bug #1019)
- A distribuição binária agora incluem a biblioteca do servidor embutido (`libmysqld`) por padrão. Devido a problemas de ligação com compiladores diferentes do gcc, ele não estava incluído em todos os pacotes da distribuição inicial da versão 4.0.15. Os pacotes afetados forma reconstruidos e distribuidos como 4.0.15a. Veja Seção 1.5.1.2 [Nutshell Embedded MySQL], Página 24.
- O MySQL agora pode usar o otimizador de faixa para `BETWEEN` com limites não constantes. (Bug #991)
- Mensagens de erro de replicação agora incluem o banco de dados padrão, assim os usuários podem verificar em qual banco de dados a consulta com erro está rodando.
- Uma alteração da documentação: Adicionado um parágrafo sobre como as opções `binlog-do-db` e `binlog-ignore-db` são testadas em um banco de dados no master (veja Seção 4.10.4 [Binary log], Página 374), e um parágrafo sobre como `replicate-do-db`, `replicate-do-table` e opções análogas são testadas em bancos de dados e tabelas no slave (veja Seção 4.11.6 [Replication Options], Página 392).
- Agora o slave não replica `SET PASSWORD` se estiver configurado para excluir o banco de dados `mysql` da replicação (usando, por exemplo, `replicate-wild-ignore-table=mysql.%`). Este já era o caso para `GRANT` e `REVOKE` desde a versão 4.0.13 (embora houvesse o Bug #980 nas versões 4.0.13 & 4.0.14, que foi corrigido na versão 4.0.15).
- Rewrote the information shown in the `State` column of `SHOW PROCESSLIST` for replication threads and for `MASTER_POS_WAIT()` and added the most common states for these threads to the documentation, veja Seção 4.11.3 [Replication Implementation Details], Página 381.
- Adiciona um teste na replicação para detectar o caso no qual o master morre no meio da gravação de uma transação no log binário; tal transação inacabada agora dispara uma mensagem de erro no slave.
- Um comando `GRANT` que cria um usuário anônimo (isto é, uma conta com nome de usuário vazio) não exige mais `FLUSH PRIVILEGES` para a conta ser conhecida no servidor. (Bug #473)

- `CHANGE MASTER` agora descarrega o '`relay-log.info`'. Anteriormente isto era feito na próxima execução de `START SLAVE`, assim se o `mysqld` fosse desligado no slave depois de `CHANGE MASTER` sem executar `START SLAVE`, o nome e posição do relay log eram perdidos. Na reinicialização eles eram carregados a partir do '`relay-log.info`', revertendo-os para seus valores antigos (incorretos) de antes do `CHANGE MASTER`, exibindo mensagens de erro (já que o relay log antigo não existia mais) e as threads slaves se recusavam a iniciar. (Bug #858)

Bugs corrigidos:

- Corrigido o overflow do buffer no tratamewnto de senhas, que podia potencialmente ser explorardo pelo usuário MySQL com privilégios na tabela `mysql.user` para executar código aleatórios para obter acessi com o UID do processo mysqld (obrgado a Jedi/Sector One por detectar e reportar este erro.)
- Corrigido um falha do servidor com `FORCE INDEX` em uma consulta contendo `"Range checked for each record"` na saída do `EXPLAIN`. (Bug #1172)
- Corrigido o tratamento de permissão de tabelas/colunas - a ordenação apropriada (do mais específico para o menos específico, veja Seção 4.3.10 [Request access], Página 242) não era respeitada (Bug #928)
- Corrigido um bug raro no MYISAM introduzido na versão 4.0.3 onde o handler do arquivo de índice não era diretamente atualizado depois de um `UPDATE` de registros dinamicos separados.
- Corrigido o erro `Can't unlock file` ao executar `myisamchk --sort-index` no Windows. (Bug #1119)
- Corrigido um possível deadlock ao alterar `key_buffer_size` enquanto a cache de chaves era ativamente usada. (Bug #1088)
- Corrigido um bug de overflow em `MyISAM` e `ISAM` quando um registro era atualiado na tabela com um grande número de colunas e pelo meno uma coluna `BLOB/TEXT`.
- Corrigido um resultado incorreto ao fazer `UNION` e `LIMIT #,#` quando não era usado parenteses na parte `SELECT`.
- Corrigido um resultado incorreto ao fazer `UNION` e `ORDER BY .. LIMIT #` quando não usado parenteses na parte `SELECT`.
- Corrigido um problema com `SELECT SQL_CALC_FOUND_ROWS ... UNION ALL ... LIMIT #` onde `FOUND_ROWS()` retornava o número incorreto de linhas.
- Corrigidos um erro de pilha indesejado quando tinhamos uma grande expressão do tipo `1+1-1+1-1...` de uma ceta combinação. (Bug #871)
- Corrigido o erro que algumas vezes fazia uma tabela com um índice `FULLTEXT` estar marcada como `"analyzed"`.
- Corrigido o MySQL para que o tamanho do campo (na API C) para a segunda coluna em `SHOW CREATE TABLE` seja sempre maior que o tamanho do dado. A única aplicação conhecida que era afetada pelo comportamento anterior era o Borland dbExpress, que truncava a saída do comando. (Bug #1064)
- Corrigida a falha na comparação de strings usando o conjunto de caracteres `tis620`. (Bug #1116)
- Corrigido um bug do `ISAM` na otimização de `MAX()`.

- `myisamchk --sort-records=N` não marca mais a tabela como danificada se a ordenação falhar devido a uma chave inapropriada. (Bug #892)
- Corrigido um erro no tratamento de tabelas `MyISAM` compactadas que algumas vezes torna impossível se reparar tabelas compactadas no modo "Repair by sort". "Repair with keycache" (`myisamchk --safe-recover`) funcionad. (Bug #1015)
- Correção de um erro na propagação do número da versão do manual incluído no arquivo de distribuição. (Bug #1020)
- Corrigida um problema de ordenacao da chave (uma chave primária - `PRIMARY` - declarada em uma coluna que não é explicitamente marcada como `NOT NULL` era ordenada depois de uma chave `UNIQUE` para uma coluna `NOT NULL`).
- Corrigido o resultado de `INTERVAL` qaundo aplicado a um valor `DATE`. (Bug #792)
- Corrida a compilação da biblioteca do servidor embutido da arquivo de especificação do RPM. (Bug #959)
- Adicionado alguns arquivos que faltavam na arquivo de especificação do RPM e corrigido alguns erros de criação do RPM que ocorriam no Red Hat Linux 9. (Bug #998)
- Corrigida a avaliação incorreta de `XOR` na cláusula `WHERE`. (Bug #992)
- Corrigido um erro com processamento na cache de consultas com tabelas unidas a partir de mais de 255 tabelas. (Bug #930)
- Correção dos resultados incorretos da consulta outer join (ex. `LEFT JOIN`) quando a condição `ON` é sempre falsa, e a faixa de busca é usada. (Bug #926)
- Corrigido um erro causando resultados incorretos de `MATCH ... AGAINST()` em algumas joins. (Bug #942)
- Tabelas `MERGE` não ignoram mais "Using index" (da saída de `EXPLAIN`).
- Corrigido um erro que fazia uma tabela vazia ser marcada como "analyzed". (Bug #937)
- Corrigida a falha em `myisamchk --sort-records` quando usada em tabelas compactadas.
- Corrigido o `ALTER TABLE` lento (quando comparado a versão 3.23) e comandos relacionados tais como `CREATE INDEX`. (Bug #712)
- Correção de segmentation fault resultante de `LOAD DATA FROM MASTER` quando o mestre estava executando sem a opção `--log-bin`. (Bug #934)
- Corrigido um erro de segurança: Um servidor compilado com suporte a SSL ainda permitia conexões por usuários que tinham a opção `REQUIRE SSL` especificadas por suas contas.
- Corrigido um erro aleatório: Algumas vezes o slave replicava consultas `GRANT` ou `REVOKE` mesmo se estivesse configurado para excluir o banco de dados `mysql` da replicação (por exemplo, usando `replicate-wild-ignore-table=mysql.%`). (Bug #980)
- Os campos `Last_Errno` e `Last_Error` na saída de `SHOW SLAVE STATUS` agora são limpadas por `CHANGE MASTER` e quando a thread slave de SQL inicia. (Bug #986)
- Um erro de documentação: ela dizia que `RESET SLAVE` não altera a informação de conexão (master host, port, user e password), embora ela o fizesse. A instrução retorna estes valores para a opção de inicialização (`master-host` etc) se houvesse alguma. (Bug #985)

- `SHOW SLAVE STATUS` agora exibe a informação correta (master host, port, user e password) depois de `RESET SLAVE` (isto é, ela mostra os novos valores, que são copiados das opções de inicialização se houver alguma). (Bug #985)
- Disabilitada a propagação da posição original do log do master para eventos porque isto gerava valores inesperados para `Exec_Master_Log_Pos` e problemas com `MASTER_POS_WAIT()` em configurações de replicação A->B->C. (Bug #1086)
- Corrigido uma segmentation fault no `mysqlbinlog` quando `--position=x` era usado com `x` estando entre um evento `Create_file` e o evento `Append_block`, `Exec_load` ou `Delete_file`. (Bug #1091)
- `mysqlbinlog` exibia avisos superfluos quando se usava `--database`, o que causava erro de sintaxe quando enviado para `mysql`. (Bug #1092)
- O `mysqlbinlog --database` também filtra `LOAD DATA INFILE` (anteriormente, ele filtrava todas as consultas exceto `LOAD DATA INFILE`). (Bug #1093)
- O `mysqlbinlog` em alguns casos esquece de colocar um `'#'` em frente do `LOAD DATA INFILE` original (este comando é exibido apenas para informação, não para ser executado; mais tarde ele funcionava como `LOAD DATA LOCAL` com um nome de arquivo diferente, para execução pelo `mysql`). (Bug #1096)
- `binlog-do-db` e `binlog-ignore-db` filtravam `LOAD DATA INFILE` incorretamente (ele era escrito parcialmente para o log binário). Isto resultava em um corrompimento do log binário, que podia fazer o slave parar com um erro. (Bug #1100)
- Quando, em uma transação, um tabela transacional (como uma tabela `InnoDB`) era atualizada, e posteriormente na mesma transação um tabela não transacional (como um tabela `MyISAM`) era atualizada usando o conteúdo atualizado da tabela transacional (com `INSERT ... SELECT` por exemplo), as consultas eram escritas no log binário em uma ordem incorreta. (Bug #873)
- Quando em uma transação, `INSERT ... SELECT` atualizava uma tabela não transacional, e um `ROLLBACK` era executado, nenhum erro era atualizado para o cliente. Agora o cliente é avisado que não se pode fazer roll back de algumas alterações, como já era o caso para um `INSERT` normal. (Bug #1113)
- Corrigido um erro portencial: Quando `STOP SLAVE` era executado enquanto a thread slave de SQL estava no meio de uma transação, e então `CHANGE MASTER` era usado para direcionar para o slave para alguma instrução não transacional, a thread slave de SQL ficava confusa (porque ela ainda podia achar que estava em uma transação).

D.3.4 Alterações na distribuição 4.0.14 (18 Jul 2003)

Funcionalidades adicionadas ou alteradas:

- `InnoDB` agora suporta indexação pelo prefixo de um campo. Isto significa, em particularm que as colunas `BLOB` e `TEXT` pode ser indexadas em tabelas `InnoDB`, o que não era possível antes.
- Uma alteração de documentação: Função `INTERVAL(NULL, ...)` retorna `-1`.
- Habilitado o `INSERT` do `SELECT` quando a tabela na qual os registros são inseridos também é uma tabela listada no `SELECT`.
- Permite `CREATE TABLE` e `INSERT` de qualquer `UNION`.

- A opção `SQL_CALC_FOUND_ROWS` agora sempre retorna o número total de rgistro de qulquer `UNION`.
- Removida a opção `--table` de `mysqlbinlog` para evitar repetir a funcionalidade `mysqldump`.
- Alterado levemente o otimizador para preferir busca de índice sobre busca em toda a tabela em alguns casos limites.
- Adicionado uma variável especifica da thread, `max_seeks_for_key`, que pode ser usada para forçar a otimização para usar chaves em vez de varrer a tabela, mesmo se a cardinalidade do índice for baixa.
- Adicionada a otimização que converte `LEFT JOIN` para joins normais em alguns casos.
- Uma alteração da documentação: adicionado um parágrafo sobre falhas em replicação (como usar um slave sobrevivente como um novo master, como resumir a configuração original). Veja Seção 4.11.9 [Replication FAQ], Página 410.
- Uma alteração de documentação: adicionado avisos sobre uso seguro do comando `CHANGE MASTER`. Veja Seção 4.11.8.1 [CHANGE MASTER TO], Página 402.
- O MySQL agora envia um aviso (e não um erro, como na versão 4.0.13) quando ele abre uma tabela que foi criada com o MySQL 4.1.
- Adicionada a opção `--nice` para `mysqld_safe` para permitir configurar a exatidão do processo `mysqld`. (Obrigado a Christian Hammers por fornecer o patch inicial.) (Bug #627)
- Adicionada a opção `--read-only` para que o `mysqld` não permita atualizações, exceto da thread escrava ou de usuários com o privilégio `SUPER`. (Pacth original de Markus Benning).
- `SHOW BINLOG EVENTS FROM x` onde `x` é menor que 4, agora converte silenciosamente `x` para 4 em vez de exibir um erro. A mesma alteração foi feita para `CHANGE MASTER TO MASTER_LOG_POS=x` e `CHANGE MASTER TO RELAY_LOG_POS=x`.
- `mysqld` agora só adiciona um tratamento de interrupção para o sinal `SIGINT` se você começá-lo com a nova opção `--gdb`. Isto é porque alguns usuários MySQL encontraram alguns problemas estranhos quando acidentalmente enviavam `SIGINT` para a threads `mysqld`.
- `RESET SLAVE` agora limpa os campos `Last_Errno` e `Last_Error` na saída de `SHOW SLAVE STATUS`.
- Adicionada a variável `max_relay_log_size`; o relay log será rotacionado automaticamente quando seu tamanho exceder `max_relay_log_size`. Mas se `max_relay_log_size` for 0 (o padrão), `max_binlog_size` será usado (como em versões mais antigas). `max_binlog_size` ainda se aplica a logs binários em qualquer caso de uso.
- `FLUSH LOGS` agora rotaciona os relay logs em adição aos outros tipos de logs que ele já rotacionava.

Bugs corrigidos:

- Comparação/ordenação para o conjunto de caracteres `latin1_de` foi reescrita. O algoritmo antigo não podia tratar casos como `"sä" > "ßa"`. Veja Seção 4.7.1.1 [German character set], Página 326. Em casos raros ela resultava em tabela corrompida.
- Corrigido um problema com a prompt de senha no Windows. (Bug #683)

- `ALTER TABLE ... UNION=(...)` para uma tabela `MERGE` agora é permitida mesmo que alguma tabela `MyISAM` seja somente leitura. (Bug #702)
- Corrigido um problema com `CREATE TABLE t1 SELECT x'41'`. (Bug #801)
- Removido alguns avisos de lock incorretos do log de erro.
- Corrigida um estouro de memória ao se fazer `REPAIR` em uma tabela com uma chave auto incremento multi-partes onde uma parte era um pacote `CHAR`.
- Corrigida uma provável condição de corrida no código da replicação que podia levar potencialmente a instruções `INSERT` não sendo replicadas no evento de um comando `FLUSH LOGS` ou quando o log binário excede `max_binlog_size`. (Bug #791)
- Corrigido um bug que pode levar a falha em `INTERVAL` e `GROUP BY` ou `DISTINCT`. (Bug #807)
- Corrigido um bug no `mysqlhotcopy`, assim ele agora aborta em operações de cópia de tabelas sem sucesso. Corrigido outro bug, assim ele obtem sucesso quando houver milhares de tabelas para copiar. (Bug #812)
- Corrigido o problema com `mysqlhotcopy` que falhava ao ler opções do arquivo de opção. (Bug #808)
- Corrigido um bug no otimizador que algumas vezes prevenia o MySQL de usar índices `FULLTEXT` mesmo se fosse possível (por exemplo, em `SELECT * FROM t1 WHERE MATCH a,b AGAINST("index") > 0`).
- Corrigido um bug com "table is full" em operações `UNION`.
- Corrigido um problema de segurança no qual usuários habilitados sem privilégios obtinham informações na lista de banco de dados existentes usando `SHOW TABLES` e comandos parecidos.
- Corrigido um problema de pilha no UnixWare/OpenUnix.
- Corrigido um problema de configuração UnixWare/OpenUNIX e OpenServer.
- Corrigido um problema de pilha cheia na verificação da senha.
- Corrigido um problema com `max_user_connections`.
- `HANDLER` sem um índice agora funciona apropriadamente quando uma tabela tem registros deletados. (Bug #787)
- Corrigido um erro com `LOAD DATA` em `mysqlbinlog`. (Bug #670)
- Correção: `SET CHARACTER SET DEFAULT` fucniona. (Bug #462)
- Corrigido o comportamento de tabelas `MERGE` em consultas `ORDER BY ... DESC`. (Bug #515)
- Corrigida a falha do servidor em `PURGE MASTER LOGS` ou `SHOW MASTER LOGS` quando o log binário estava desligado. (Bug #733)
- Corrigido o problema de verificação de senha no Windows. (Bug #464)
- Corrigido um erro na comparação de uma coluna `DATETIME` e uma constante inteira. (Bug #504)
- Corrigido o modo remoto de `mysqlbinlog`. (Bug #672)
- Corrigido `ERROR 1105: Unknown error` que ocorria para algumas consultas `SELECT`, onde uma coluna declarada como `NOT NULL` era comparada com uma expressão que podia tomar o valor `NULL`.

- Alterado o timeout em `mysql_real_connect()` para usar `poll()` em vez de `select()` para contornar problemas cmo muitos outros arquivos abertos no cliente.
- Corrigido resultados incorretos de `MATCH ... AGAINST` usado com uma consulta `LEFT JOIN`.
- Corrigido um bug que limitava o valor máximo para variáveis `mysqld` em 4294967295 quando eles eram especificados na linha de comando.
- Corrigido um bug que algumas vezes causavam falsos erros de "Access denied" nas instruções `HANDLER ... READ`, quando uma tabela é referenciada via um alias.
- Corrigido um problema de portabilidade com `safe_malloc`, o qual fazia com que o MySQL para enviar erros de "Freeing wrong aligned pointer" no SCO 3.2.
- `ALTER TABLE ... ENABLE/DISABLE KEYS` podia causar um core dump quando feito depois de uma instrução `INSERT DELAYED` na mesma tabela.
- Corrigido um problema com conversão da hora local para GMT onde algumas vezes resultava em diferentes (mas corretos) timestamps. Agora o MySQL deve usar o menor valor de possível neste caso. (Bug #316)
- Uma cache de consultas muito pequena podia fazer o `mysqld` falhar. (Bug #549)
- Corrigido um bug (acidentalemnte introduzida por nós mas presente apenas na versão 4.0.13) que faz `INSERT ... SELECT` em uma coluna `AUTO_INCREMENT` que não replica bem. Este bug está no master, não no slave. (Bug #490)
- Corrigido um bug: Quando uma instrução `INSERT ... SELECT` inseria linhas em uma tabela não transacional, mas falhava no mesmo ponto (por exemplo, devido a erros de "Duplicate key"), a consulta não era escrita no log binário. Agora ela é escrita no log binário, com seus códigos de erros, como todas as outras cosultas são. Sobre a opção `slave-skip-errors` para como tratar consultas completadas parcialmente no slave, veja Seção 4.11.6 [Replication Options], Página 392. (Bug #491)
- `SET FOREIGN_KEY_CHECKS=0` não era replicado apropriadamente. A correção provavelmente não será feita para 3.23.
- Em um slave, `LOAD DATA INFILE` sem cláusulas `IGNORE` ou `REPLACE` no master, era replicada com `IGNORE`. Enquanto isto não for um problemase os dados do master e slave são identicos (em `LOAD` que não produz conflitos de duplicação no master não produzirá nada no slave de qualquer forma), o que é verdade em operações normais, para depuração é melhor não adicionar silenciosamente o `IGNORE`. Deste modo, você pode obter uma mensagem de erro no slave e descobrir que por alguma razão, os dados no master e slave são diferentes e investigar o porque. (Bug #571)
- Em um slave, `LOAD DATA INFILE` exibia uma mensagem incomplete "Duplicate entry '%-.64s' for key %d"' (o nome e valor da chave não eram mencionados) no caso de conflito de duplicação (o que não acontece em operações normais). (Bug #573)
- Quando usado um slave compilado com `--debug`, `CHANGE MASTER TO RELAY_LOG_POS` podia causar um falha de declaração da depuração. (Bug #576)
- Ao fazer um `LOCK TABLES WRITE` em uma tabela `InnoDB`, o commit podia não acontecer, se a consulta não era escrita no log binário (por exemplo, se `--log-bin` não era usado, ou `binlog-ignore-db` era usado). (Bug #578)
- Se um master na versão 3.23 tivesse aberto tabelas temporárias que tinham sido replicadas para um slave na versão 4.0, e o log binário rotacionado, estas tabelas temporárias

eram automaticamente removidas pelo slave (o que causa problemas se o master os utiliza subsequecialmente). Este erro foi corrigido na versão 4.0.13, mas de um modo que cria um incoveniência indesejada: se o master na versão 3.23 morrer brutalmente. (queda de força), sem tempo suficiente para escrever automaticamente instruções `DROP TABLE` em seu log binário. então o slave na versão 4.0.13 não notificaria que as tabelas temporárias tinham sido removidas, até o servidor `mysqld` slave ter sido reiniciado. Este pequeno incoveniente está corrigido na versão 3.23.57 e 4.0.14 (significando que o master deve ser atualizado para a versão 3.23.57 e o slave para a 4.0.14 para remover o incoveniente). (Bug #254)

- Se `MASTER_POS_WAIT()` estava espereando e o slave estava inativo, e thread slave de SQL terminada, `MASTER_POS_WAIT()` esperaria para sempre. Agora quando a thread slave de SQL termina, `MASTER_POS_WAIT()` retorna `NULL` imediatamente ("slave stopped"). (Bug #651)
- Depois de `RESET SLAVE; START SLAVE;`, o valor de `Relay_Log_Space` exibido por `SHOW SLAVE STATUS` era muito grande para 4 bytes. (Bug #763)
- Se uma consulta era ignorada no slave (devido a `replicate-ignore-table` e outras regras similares), o escravo ainda verifica se a consulta consegue o mesmo código de erro (0, sem erro) como no master. Assim se o master tiver um erro na consulta (por exemplo, "Duplicate entry" em uma inserção de múltiplas linhas), então o slave parava e avisava que código de erro não coincidia. (Bug #797)

D.3.5 Alterações na distribuição 4.0.13 (16 May 2003)

Funcionalidades adicionadas ou alteradas:

- `PRIMARY KEY` agora implica `NOT NULL`. (Bug #390)
- O pacote de binários do Windows agora está compilado com `--enable-local-infile` encontrar a configuraçào de construção do Unix.
- Removida a medida do tempo de `mysql-test-run`. `time` não aceita todos os parâmetros exigidos em muitas aplicações (por exemplo, QNX) e a medida de tempo não é reamente necessária (isto não é um benchmark).
- `SHOW MASTER STATUS` e `SHOW SLAVE STATUS` exigem o privilégio `SUPER`; agora eles aceitam `REPLICATION CLIENT`. (Bug #343)
- Adicionada otimização de reparação do MyISAM em multi-threads e a variável `myisam_repair_threads` para habilitá-lo. Veja Seção 4.6.8.4 [`myisam_repair_threads`], Página 309.
- Adicionada a variável `innodb_max_dirty_pages_pct` que controla a quantidade de páginas "sujas" permitidas na área de buffer do `InnoDB`.
- As mensagens de erro `CURRENT_USER()` e `Access denied` agora relatam o nome de máquina exatamente como ele está especificado no comando `GRANT`.
- Removido os resultados de benchmark das distribuições fonte e binárias. Eles ainda estão disponíveis na árvore fonte do BK.
- Tabelas `InnoDB` agora suportam `ANALYZE TABLE`.
- O MySQL agora envia um erro quando ele abre uma tabela que foi criada com o MySQL 4.1.

- A opção `--new` agora altera altera os itens binários (`0xFFDF`) para serem tratados como strings binárias em vez de números por padrão. Isto corrige alguns problemas com conjunto de caracteres onde é conveniente colocar a string como um item binário. Depois destas alterações você deve converter a string binária para `INTEGER` com um `CAST` se você quiser comparar dois itens binários, um com o outro, e saber qual é maior. `SELECT CAST(0xfeff AS UNSIGNED) < CAST(0xff AS UNSIGNED)`. Este será o comportamento padrão no MySQL 4.1. (Bug #152)
- Habilitado `delayed_insert_timeout` no Linux (as bibliotecas glibc mais modernas tem um `pthread_cond_timedwait` corrigido). (Bug #211)
- Não cria mais threads de insert delayed que o dado por `max_insert_delayed_threads`. (Bug #211)
- Alterado o `UPDATE ... LIMIT` para aplicar o limite as linhas encontradas, independente de terem sido alteradas. Anteriormente o limite era aplicado como uma restrição no número de linhas alteradas.
- Ajustado o otimizador para favorecer indíces em cluster em ver de busca na tabela.
- `BIT_AND()` e `BIT_OR()` agora retornam um valor de 64 bits sem sinal.
- Adicionado avisos ao log de erro do porquê de um falha em uma conexão segura (quando executando com `--log-warnings`).
- As opções `--skip-symlink` e `--use-symbolic-links` estão obsoletas e forma substituídas com `--symbolic-links`.
- A opção padrão para `innodb_flush_log_at_trx_commit` foi alterada de 0 para 1 para tornar tabelas `InnoDB` como ACID por padrão. Veja Seção 7.5.3 [`InnoDB` start], Página 644.
- Adicionado o recurso para `SHOW KEYS` para mostrar chaves que estão disabilitadas pelo comando `ALTER TABLE DISABLE KEYS`.
- Ao usar um tipo de tabela não existente com `CREATE TABLE`, primeiro vê se o tipo de tabela padrão existe antes de utilizar `MyISAM`.
- Adicionado `MEMORY` como um alias para `HEAP`.
- Renomeada a função `rnd` para `my_rnd` já que o nome era muito genérico e é um símbolo exportado no `libmysqlclient` (obrigado a Dennis Haney pelo patch inicial).
- Correção de portabilidade: renomeado '`include/dbug.h`' para '`include/my_debug.h`'.
- `mysqldump` não deleta mais o log binário sem aviso quando chamado com `--master-data` ou `--first-slave`; enquanto este comportamento era conveniente para alguns usuários, outros podia sofrer com ele. Agora deve perguntar explicitamente pela sua deleção com a nova opção `--delete-master-logs`.
- Se o slave é configurado (usando, por exemplo, `replicate-wild-ignore-table=mysql.%`) para ecluir `mysql.user`, `mysql.host`, `mysql.db`, `mysql.tables_priv` e `mysql.columns_priv` da replicação, então `GRANT` e `REVOKE` não serão replicados.

Bugs corrigidos:

- A mensagem de erro `Access denied` ao logar tinha um valor `Using password` incorreto. (Bug #398)
- Corrigido um bug com `NATURAL LEFT JOIN`, `NATURAL RIGHT JOIN` e `RIGHT JOIN` quando usadas muitas tabelas em joins. O problema era que o método `JOIN` não era sempre

associoado com as tabelas envolvida no método `JOIN`. Se você tiver uma consulta que usa muitos `RIGHT JOIN` ou `NATURAL ... JOINS` você deve verificar se eles funcionam como você espera depois de atualizar o MySQL para esta versõa. (Bug #291)

- O cliente de linha de comando `mysql` não olha mais os comnados `\*` dentro de stringd com aspas invertidas.
- Corrigido `Unknown error` ao usar `UPDATE ... LIMIT`. (Bug #373)
- Corrigido o problema com o modo ANSI e `GROUP BY` com constantes. (Bug #387)
- Corrigido o erro com `UNION` e `OUTER JOIN`. (Bug #386)
- Corrigido o erro se é usado um `UPDATE` multi-tabelas e a consulta exige um tabela temporária maior que `tmp_table_size`. (Bug #286)
- Executa `mysql_install_db` com a opção `-IN-RPM` para a instalação do Mac OS X não falhar em sistemas com a configuração de nome de máquina feita de forma inapropriada.
- `LOAD DATA INFILE` agora irá ler `000000` como uma data zerada em vez de `"2000-00-00"`.
- Corrigido um erro que fazia que `DELETE FROM table WHERE const_expression` sempre deletasse toda a tabela (mesmo se o resultado da expressão fosse falso). (Bug #355)
- Corrigido um bug de core dump ao usar `FORMAT('nan',#)`. (Bug #284)
- Corrigido um erro na resolução do nome com `HAVING ... COUNT(DISTINCT ...)`.
- Corrigido resultados incorretos da operação de truncamento (`*`) em `MATCH ... AGAINST()` em alguns joins complexos.
- Fixed a crash in `REPAIR ... USE_FRM` command, when used on read-only, nonexisting table or a table with a crashed index file.
- Fixed a crashing bug in mysql monitor program. It occurred if program was started with `--no-defaults`, with a prompt that contained hostname and connection to non-existing db was requested
- Fixed problem when comparing a key for a multi-byte-character set. (Bug #152)
- Fixed bug in `LEFT`, `RIGHT` and `MID` when used with multi-byte character sets and some `GROUP BY` queries. (Bug #314)
- Fix problem with `ORDER BY` being discarded for some `DISTINCT` queries. (Bug #275)
- Fixed that `SET SQL_BIG_SELECTS=1` works as documented (This corrects a new bug introduced in 4.0)
- Fixed some serious bugs in `UPDATE ... ORDER BY`. (Bug #241)
- Fixed unlikely problem in optimising `WHERE` clause with constant expression like in `WHERE 1 AND (a=1 AND b=1)`.
- Fixed that `SET SQL_BIG_SELECTS=1` works again.
- Introduced proper backtick quoting for db.table in `SHOW GRANTS`.
- `FULLTEXT` index stopped working after `ALTER TABLE` that converts `TEXT` column to `CHAR`. (Bug #283)
- Fixed a security problem with `SELECT` and wildcarded select list, when user only had partial column `SELECT` privileges on the table.
- Mark a MyISAM table as "analyzed" only when all the keys are indeed analyzed.

- Only ignore world-writeable '`my.cnf`' files that are regular files (and not, for example, named pipes or character devices).
- Fixed few smaller issues with `SET PASSWORD`.
- Fixed error message which contained deprecated text.
- Fixed a bug with two `NATURAL JOIN`s in the query.
- `SUM()` didn't return `NULL` when there was no rows in result or when all values was `NULL`.
- On Unix symbolic links handling was not enabled by default and there was no way to turn this on.
- Added missing dashes to parameter `--open-files-limit` in `mysqld_safe`. (Bug #264)
- Fixed incorrect hostname for TCP/IP connections displayed in `SHOW PROCESSLIST`.
- Fixed a bug with `NAN` in `FORMAT(...)` function ...
- Fixed a bug with improperly cached database privileges.
- Fixed a bug in `ALTER TABLE ENABLE / DISABLE KEYS` which failed to force a refresh of table data in the cache.
- Fixed bugs in replication of `LOAD DATA INFILE` for custom parameters (`ENCLOSED`, `TERMINATED` and so on) and temporary tables. (Bug #183, Bug #222)
- Fixed a replication bug when the master is 3.23 and the slave 4.0: the slave lost the replicated temporary tables if `FLUSH LOGS` was issued on the master. (Bug #254)
- Fixed a bug when doing `LOAD DATA INFILE IGNORE`: When reading the binary log, `mysqlbinlog` and the replication code read `REPLACE` instead of `IGNORE`. This could make the slave's table become different from the master's table. (Bug #218)
- Fixed a deadlock when `relay_log_space_limit` was set to a too small value. (Bug #79)
- Fixed a bug in HAVING clause when an alias is used from the **select list**.
- Fixed overflow bug in `MyISAM` when a row is inserted into a table with a large number of columns and at least one `BLOB/TEXT` column. Bug was caused by incorrect calculation of the needed buffer to pack data.
- Fixed a bug when `SELECT` **@nonexistent_variable** caused the error in client - server protocol due to net_printf() being sent to the client twice.
- Fixed a bug in setting `SQL_BIG_SELECTS` option.
- Fixed a bug in `SHOW PROCESSLIST` which only displayed a localhost in the `"Host"` column. This was caused by a glitch that only used current thread information instead of information from the linked list of threads.
- Removed unnecessary Mac OS X helper files from server RPM. (Bug #144)
- Allow optimization of multiple-table update for `InnoDB` tables as well.
- Fixed a bug in multiple-table updates that caused some rows to be updated several times.
- Fixed a bug in `mysqldump` when it was called with `--master-data`: the `CHANGE MASTER TO` commands appended to the SQL dump had incorrect coordinates. (Bug #159)
- Fixed a bug when an updating query using `USER()` was replicated on the slave; this caused segfault on the slave. (Bug #178). `USER()` is still badly replicated on the slave (it is replicated to `""`).

D.3.6 Alterações na distribuição 4.0.12 (15 Mar 2003: Production)

Functionality added or changed:

- `mysqld` no longer reads options from world-writeable config files.
- Integer values between 9223372036854775807 and 9999999999999999999 are now regarded as unsigned longlongs, not as floats. This makes these values work similar to values between 10000000000000000000 and 18446744073709551615.
- `SHOW PROCESSLIST` will now include the client TCP port after the hostname to make it easier to know from which client the request originated.

Bugs fixed:

- Fixed `mysqld` crash on extremely small values of `sort_buffer` variable.
- `INSERT INTO u SELECT ... FROM t` was written too late to the binary log if t was very frequently updated during the execution of this query. This could cause a problem with `mysqlbinlog` or replication. The master must be upgraded, not the slave. (Bug #136)
- Fixed checking of random part of `WHERE` clause. (Bug #142)
- Fixed a bug with multiple-table updates with `InnoDB` tables. This bug occurred as, in many cases, `InnoDB` tables cannot be updated "on the fly," but offsets to the records have to be stored in a temporary table.
- Added missing file `mysql_secure_installation` to the `server` RPM subpackage. (Bug #141)
- Fixed MySQL (and `myisamchk`) crash on artificially corrupted `.MYI` files.
- Don't allow `BACKUP TABLE` to overwrite existing files.
- Fixed a bug with multi-table `UPDATE` statements when user had all privileges on the database where tables are located and there were any entries in `tables_priv` table, that is, `grant_option` was true.
- Fixed a bug that allowed a user with table or column grants on some table, `TRUNCATE` any table in the same database.
- Fixed deadlock when doing `LOCK TABLE` followed by `DROP TABLE` in the same thread. In this case one could still kill the thread with `KILL`.
- `LOAD DATA LOCAL INFILE` was not properly written to the binary log (hence not properly replicated). (Bug #82)
- `RAND()` entries were not read correctly by `mysqlbinlog` from the binary log which caused problems when restoring a table that was inserted with `RAND()`. `INSERT INTO t1 VALUES(RAND())`. In replication this worked ok.
- `SET SQL_LOG_BIN=0` was ignored for `INSERT DELAYED` queries. (Bug #104)
- `SHOW SLAVE STATUS` reported too old positions (columns `Relay_Master_Log_File` and `Exec_Master_Log_Pos`) for the last executed statement from the master, if this statement was the `COMMIT` of a transaction. The master must be upgraded for that, not the slave. (Bug #52)
- `LOAD DATA INFILE` was not replicated by the slave if `replicate_*_table` was set on the slave. (Bug #86)

- After `RESET SLAVE`, the coordinates displayed by `SHOW SLAVE STATUS` looked un-reset (though they were, but only internally). (Bug #70)
- Fixed query cache invalidation on `LOAD DATA`.
- Fixed memory leak on `ANALYZE` procedure with error.
- Fixed a bug in handling `CHAR(0)` columns that could cause incorrect results from the query.
- Fixed rare bug with incorrect initialisation of `AUTO_INCREMENT` column, as a secondary column in a multi-column key (veja Seção 3.6.9 [`AUTO_INCREMENT` on secondary column in a multi-column key], Página 201), when data was inserted with `INSERT ... SELECT` or `LOAD DATA` into an empty table.
- On Windows, `STOP SLAVE` didn't stop the slave until the slave got one new command from the master (this bug has been fixed for MySQL 4.0.11 by releasing updated 4.0.11a Windows packages, which include this individual fix on top of the 4.0.11 sources). (Bug #69)
- Fixed a crash when no database was selected and `LOAD DATA` command was issued with full table name specified, including database prefix.
- Fixed a crash when shutting down replication on some platforms (for example, Mac OS X).
- Fixed a portability bug with `pthread_attr_getstacksize` on HP-UX 10.20 (Patch was also included in 4.0.11a sources).
- Fixed the `bigint` test to not fail on some platforms (for example, HP-UX and Tru64) due to different return values of the `atof()` function.
- Fixed the `rpl_rotate_logs` test to not fail on certain platforms (e.g. Mac OS X) due to a too long file name (changed `slave-master-info.opt` to `.slave-mi`).

D.3.7 Alterações na distribuição 4.0.11 (20 Feb 2003)

Functionality added or changed:

- `NULL` is now sorted **LAST** if you use `ORDER BY ... DESC` (as it was before MySQL 4.0.2). This change was required to comply with the SQL-99 standard. (The original change was made because we thought that SQL-99 required `NULL` to be always sorted at the same position, but this was incorrect).
- Added `START TRANSACTION` (SQL-99 syntax) as alias for `BEGIN`. This is recommended to use instead of `BEGIN` to start a transaction.
- Added `OLD_PASSWORD()` as a synonym for `PASSWORD()`.
- Allow keyword `ALL` in group functions.
- Added support for some new `INNER JOIN` and `JOIN` syntaxes. For example, `SELECT * FROM t1 INNER JOIN t2` didn't work before.
- Novell NetWare 6.0 porting effort completed, Novell patches merged into the main source tree.

Bugs fixed:

- Fixed problem with multiple-table delete and `InnoDB` tables.

- Fixed a problem with `BLOB NOT NULL` columns used with `IS NULL`.
- Re-added missing pre- and post(un)install scripts to the Linux RPM packages (they were missing after the renaming of the server subpackage).
- Fixed that table locks are not released with multi-table updates and deletes with `InnoDB` storage engine.
- Fixed bug in updating `BLOB` columns with long strings.
- Fixed integer-wraparound when giving big integer (>= 10 digits) to function that requires an unsigned argument, like `CREATE TABLE (...) AUTO_INCREMENT=#`.
- `MIN(key_column)` could in some cases return `NULL` on a column with `NULL` and other values.
- `MIN(key_column)` and `MAX(key_column)` could in some cases return incorrect values when used in `OUTER JOIN`.
- `MIN(key_column)` and `MAX(key_column)` could return incorrect values if one of the tables was empty.
- Fixed rare crash in compressed MyISAM tables with blobs.
- Fixed bug in using aggregate functions as argument for `INTERVAL`, `CASE`, `FIELD`, `CONCAT_WS`, `ELT` and `MAKE_SET` functions.
- When running with `--lower-case-table-names` (default on Windows) and you had tables or databases with mixed case on disk, then executing `SHOW TABLE STATUS` followed with `DROP DATABASE` or `DROP TABLE` could fail with `Errcode 13`.

D.3.8 Alterações na distribuição 4.0.10 (29 Jan 2003)

Functionality added or changed:

- Added option `--log-error[=file_name]` to `mysqld_safe` and `mysqld`. This option will force all error messages to be put in a log file if the option `--console` is not given. On Windows `--log-error` is enabled as default, with a default name of `host_name.err` if the name is not specified.
- Changed some things from `Warning:` to `Note:` in the log files.
- The mysqld server should now compile on NetWare.
- Added optimization that if one does `GROUP BY ... ORDER BY NULL` then result is not sorted.
- New `--ft-stopword-file` command-line option for `mysqld` to replace/disable the built-in stopword list that is used in full-text searches. Veja Seção 4.6.8.4 [ft_stopword_file], Página 309.
- Changed default stack size from 64K to 192K; This fixes a core dump problem on Red Hat 8.0 and other systems with a `glibc` that requires a stack size larger than 128K for `gethostbyaddr()` to resolve a hostname. You can fix this for earlier MySQL versions by starting `mysqld` with `--thread-stack=192K`.
- Added `mysql_waitpid` to the binary distribution and the `MySQL-client` RPM subpackage (required for `mysql-test-run`).
- Renamed the main `MySQL` RPM package to `MySQL-server`. When updating from an older version, `MySQL-server.rpm` will simply replace `MySQL.rpm`.

- If a slave is configured with `replicate_wild_do_table=db.%` or `replicate_wild_ignore_table=db.%`, these rules will be applied to `CREATE/DROP DATABASE` too.
- Added timeout value for `MASTER_POS_WAIT()`.

Bugs fixed:

- Fixed initialisation of the random seed for newly created threads to give a better `rand()` distribution from the first call.
- Fixed a bug that caused `mysqld` to hang when a table was opened with the `HANDLER` command and then dropped without being closed.
- Fixed bug in logging to binary log (which affects replication) a query that inserts a `NULL` in an `AUTO_INCREMENT` column and also uses `LAST_INSERT_ID()`.
- Fixed an unlikely bug that could cause a memory overrun when using `ORDER BY constant_expression`.
- Fixed a table corruption in `myisamchk`'s parallel repair mode.
- Fixed bug in query cache invalidation on simple table renaming.
- Fixed bug in `mysqladmin --relative`.
- On some 64 bit systems, `show status` reported a strange number for `Open_files` and `Open_streams`.
- Fixed incorrect number of columns in `EXPLAIN` on empty table.
- Fixed bug in `LEFT JOIN` that caused zero rows to be returned in the case the `WHERE` condition was evaluated as `FALSE` after reading `const` tables. (Unlikely condition).
- `FLUSH PRIVILEGES` didn't correctly flush table/column privileges when `mysql.tables_priv` is empty.
- Fixed bug in replication when using `LOAD DATA INFILE` one a file that updated an `AUTO_INCREMENT` column with `NULL` or `0`. This bug only affected MySQL 4.0 masters (not slaves or MySQL 3.23 masters). **Note**: If you have a slave that has replicated a file with generated `AUTO_INCREMENT` columns then the slave data is corrupted and you should reinitialise the affected tables from the master.
- Fixed possible memory overrun when sending a `BLOB` value larger than 16M to the client.
- Fixed incorrect error message when setting a `NOT NULL` column to an expression that returned `NULL`.
- Fixed core dump bug in `str LIKE "%other_str%"` where `str` or `other_str` contained characters >= 128.
- Fixed bug: When executing on master `LOAD DATA` and `InnoDB` failed with `table full` error the binary log was corrupted.

D.3.9 Alterações na distribuição 4.0.9 (09 Jan 2003)

Functionality added or changed:

- `OPTIMIZE TABLE` will for MyISAM tables treat all `NULL` values as different when calculating cardinality. This helps in optimising joins between tables where one of the tables has a lot of `NULL` values in a indexed column:

```
SELECT * from t1,t2 where t1.a=t2.key_with_a_lot_of_null;
```

- Added join operator `FORCE INDEX (key_list)`. This acts likes `USE INDEX (key_list)` but with the addition that a table scan is assumed to be VERY expensive. One bad thing with this is that it makes `FORCE` a reserved word.
- Reset internal row buffer in MyISAM after each query. This will reduce memory in the case you have a lot of big blobs in a table.

Bugs fixed:

- A security patch in 4.0.8 causes the mysqld server to die if the remote hostname can't be resolved. This is now fixed.
- Fixed crash when replication big `LOAD DATA INFILE` statement that caused log rotation.

D.3.10 Alterações na distribuição 4.0.8 (07 Jan 2003)

Functionality added or changed:

- Default `max_packet_length` for libmysqld.c is now 1024*1024*1024.
- One can now specify `max_allowed_packet` in a file ready by `mysql_options(MYSQL_READ_DEFAULT_FILE)`. for clients.
- When sending a too big packet to the server with the not compressed protocol, the client now gets an error message instead of a lost connection.
- We now send big queries/result rows in bigger hunks, which should give a small speed improvement.
- Fixed some bugs with the compressed protocol for rows > 16M.
- `InnoDB` tables now also support `ON UPDATE CASCADE` in `FOREIGN KEY` constraints. See the `InnoDB` section in the manual for the `InnoDB` changelog.

Bugs fixed:

- Fixed bug in `ALTER TABLE` with BDB tables.
- Fixed core dump bug in `QUOTE()` function.
- Fixed a bug in handling communication packets bigger than 16M. Unfortunately this required a protocol change; If you upgrade the server to 4.0.8 and above and have clients that uses packets >= 255*255*255 bytes (=16581375) you must also upgrade your clients to at least 4.0.8. If you don't upgrade, the clients will hang when sending a big packet.
- Fixed bug when sending blobs longer than 16M to client.
- Fixed bug in `GROUP BY` when used on BLOB column with `NULL` values.
- Fixed a bug in handling `NULL` values in `CASE` ... WHEN ...

D.3.11 Alterações na distribuição 4.0.7 (20 Dec 2002)

Functionality added or changed:

- `mysqlbug` now also reports the compiler version used for building the binaries (if the compiler supports the option `--version`).

Bugs fixed:

- Fixed compilation problems on OpenUnix and HPUX 10.20.
- Fixed some optimization problems when compiling MySQL with `-DBIG_TABLES` on a 32 bit system.
- `mysql_drop_db()` didn't check permissions properly so anyone could drop another users database. `DROP DATABASE` is checked properly.

D.3.12 Alterações na distribuição 4.0.6 (14 Dec 2002: Gamma)

Functionality added or changed:

- Added syntax support for `CHARACTER SET xxx` and `CHARSET=xxx` table options (to be able to read table dumps from 4.1).
- Fixed replication bug that caused the slave to loose its position in some cases when the replication log was rotated.
- Fixed that a slave will restart from the start of a transaction if it's killed in the middle of one.
- Moved the manual pages from '`man`' to '`man/man1`' in the binary distributions.
- The default type returned by `IFNULL(A,B)` is now set to be the more 'general' of the types of `A` and `B`. (The order is `STRING`, `REAL` or `INTEGER`).
- Moved the `mysql.server` startup script in the RPM packages from '`/etc/rc.d/init.d/mysql`' to '`/etc/init.d/mysql`' (which almost all current Linux distributions support for LSB compliance).
- Added `Qcache_lowmem_prunes` status variable (number of queries that were deleted from cache because of low memory).
- Fixed `mysqlcheck` so it can deal with table names containing dashes.
- Bulk insert optimization (veja Seção 4.6.8.4 [`bulk_insert_buffer_size`], Página 309) is no longer used when inserting small (less than 100) number of rows.
- Optimization added for queries like `SELECT ... FROM merge_table WHERE indexed_column=constant_expr`.
- Added functions `LOCALTIME` and `LOCALTIMESTAMP` as synonyms for `NOW()`.
- `CEIL` is now an alias for `CEILING`.
- The `CURRENT_USER()` function can be used to get a `user@host` value as it was matched in the `GRANT` system. Veja Seção 6.3.6.2 [`CURRENT_USER()`], Página 547.
- Fixed `CHECK` constraints to be compatible with SQL-99. This made `CHECK` a reserved word. (Checking of `CHECK` constraints is still not implemented).
- Added `CAST(... as CHAR)`.
- Added PostgreSQL compatible `LIMIT` syntax: `SELECT ... LIMIT row_count OFFSET offset`
- `mysql_change_user()` will now reset the connection to the state of a fresh connect (Ie, `ROLLBACK` any active transaction, close all temporary tables, reset all user variables etc..)

- `CHANGE MASTER` and `RESET SLAVE` now require that slave threads be both already stopped; these commands will return an error if at least one of these two threads is running.

Bugs fixed:

- Fixed number of found rows returned in `multi table updates`
- Make `--lower-case-table-names` default on Mac OS X as the default file system (HFS+) is case insensitive. Veja Seção 6.1.3 [Name case sensitivity], Página 473.
- Transactions in `AUTOCOMMIT=0` mode didn't rotate binary log.
- A fix for the bug in a `SELECT` with joined tables with `ORDER BY` and `LIMIT` clause when filesort had to be used. In that case `LIMIT` was applied to filesort of one of the tables, although it could not be. This fix also solved problems with `LEFT JOIN`.
- `mysql_server_init()` now makes a copy of all arguments. This fixes a problem when using the embedded server in C# program.
- Fixed buffer overrun in `libmysqlclient` library that allowed a malicious `MySQL` server to crash the client application.
- Fixed security-related bug in `mysql_change_user()` handling. All users are strongly recommended to upgrade to version 4.0.6.
- Fixed bug that prevented `--chroot` command-line option of `mysqld` from working.
- Fixed bug in phrase operator `"..."` in boolean full-text search.
- Fixed bug that caused `OPTIMIZE TABLE` to corrupt the table under some rare circumstances.
- Part rewrite of multi-table-update to optimise it, make it safer and more bug free.
- `LOCK TABLES` now works together with multi-table-update and multi-table-delete.
- `--replicate-do=xxx` didn't work for `UPDATE` commands. (Bug introduced in 4.0.0)
- Fixed shutdown problem on Mac OS X.
- Major `InnoDB` bugs in `REPLACE, AUTO_INCREMENT, INSERT INTO ... SELECT ...` were fixed. See the `InnoDB` changelog in the `InnoDB` section of the manual.
- `RESET SLAVE` caused a crash if the slave threads were running.

D.3.13 Alterações na distribuição 4.0.5 (13 Nov 2002)

Functionality added or changed:

- Port number was added to host name (if it is known) in `SHOW PROCESSLIST` command
- Changed handling of last argument in `WEEK()` so that one can get week number according to the ISO 8601 specification. (Old code should still work).
- Fixed that `INSERT DELAYED` threads doesn't hang on `Waiting for INSERT` when one sends a `SIGHUP` to mysqld.
- Change that `AND` works according to SQL-99 when it comes to `NULL` handling. In practice, this only affects queries where you do something like `WHERE ... NOT (NULL AND 0)`.

- `mysqld` will now resolve `basedir` to its full path (with `realpath()`). This enables one to use relative symlinks to the MySQL installation directory. This will however cause `show variables` to report different directories on systems where there is a symbolic link in the path.
- Fixed that MySQL will not use index scan on index disabled with `IGNORE INDEX` or `USE INDEX`. to be ignored.
- Added `--use-frm` option to `mysqlcheck`. When used with `REPAIR`, it gets the table structure from the `.frm` file, so the table can be repaired even if the `.MYI` header is corrupted.
- Fixed bug in `MAX()` optimization when used with `JOIN` and `ON` expressions.
- Added support for reading of MySQL 4.1 table definition files.
- `BETWEEN` behaviour changed (veja Seção 6.3.1.2 [Comparison Operators], Página 504). Now `datetime_col BETWEEN timestamp AND timestamp` should work as expected.
- One can create `TEMPORARY MERGE` tables now.
- `DELETE FROM myisam_table` now shrinks not only the '`.MYD`' file but also the '`.MYI`' file.
- When one uses the `--open-files-limit=#` option to `mysqld_safe` it's now passed on to `mysqld`.
- Changed output from `EXPLAIN` from `'where used'` to `'Using where'` to make it more in line with other output.
- Removed variable `safe_show_database` as it was no longer used.
- Updated source tree to be built using `automake 1.5` and `libtool 1.4`.
- Fixed an inadvertently changed option (`--ignore-space`) back to the original `--ignore-spaces` in `mysqlclient`. (Both syntaxes will work).
- Don't require `UPDATE` privilege when using `REPLACE`.
- Added support for `DROP TEMPORARY TABLE ...`, to be used to make replication safer.
- When transactions are enabled, all commands that update temporary tables inside a `BEGIN/COMMIT` are now stored in the binary log on `COMMIT` and not stored if one does `ROLLBACK`. This fixes some problems with non-transactional temporary tables used inside transactions.
- Allow braces in joins in all positions. Formerly, things like `SELECT * FROM (t2 LEFT JOIN t3 USING (a)), t1` worked, but not `SELECT * FROM t1, (t2 LEFT JOIN t3 USING (a))`. Note that braces are simply removed, they do not change the way the join is executed.
- `InnoDB` now supports also isolation levels `READ UNCOMMITTED` and `READ COMMITTED`. For a detailed `InnoDB` changelog, see Seção 7.5.16 [InnoDB change history], Página 680 in this manual.

Bugs fixed:

- Fixed bug in `MAX()` optimization when used with `JOIN` and `ON` expressions.
- Fixed that `INSERT DELAY` threads don't hang on `Waiting for INSERT` when one sends a `SIGHUP` to `mysqld`.
- Fixed that MySQL will not use an index scan on an index that has been disabled with `IGNORE INDEX` or `USE INDEX`.

- Corrected test for `root` user in `mysqld_safe`.
- Fixed error message issued when storage engine cannot do `CHECK` or `REPAIR`.
- Fixed rare core dump problem in complicated `GROUP BY` queries that didn't return any result.
- Fixed `mysqlshow` to work properly with wildcarded database names and with database names that contain underscores.
- Portability fixes to get MySQL to compile cleanly with Sun Forte 5.0.
- Fixed `MyISAM` crash when using dynamic-row tables with huge numbers of packed fields.
- Fixed query cache behaviour with `BDB` transactions.
- Fixed possible floating point exception in `MATCH` relevance calculations.
- Fixed bug in full-text search `IN BOOLEAN MODE` that made `MATCH` to return incorrect relevance value in some complex joins.
- Fixed a bug that limited `MyISAM` key length to a value slightly less that 500. It is exactly 500 now.
- Fixed that `GROUP BY` on columns that may have a `NULL` value doesn't always use disk based temporary tables.
- The filename argument for the `--des-key-file` argument to `mysqld` is interpreted relative to the data directory if given as a relative pathname.
- Removed a condition that temp table with index on column that can be `NULL` has to be `MyISAM`. This was okay for 3.23, but not needed in 4.*. This resulted in slowdown in many queries since 4.0.2.
- Small code improvement in multi-table updates.
- Fixed a newly introduced bug that caused `ORDER BY ... LIMIT row_count` to not return all rows.
- Fixed a bug in multi-table deletes when outer join is used on an empty table, which gets first to be deleted.
- Fixed a bug in multi-table updates when a single table is updated.
- Fixed bug that caused `REPAIR TABLE` and `myisamchk` to corrupt `FULLTEXT` indexes.
- Fixed bug with caching the `mysql` grant table database. Now queries in this database are not cached in the query cache.
- Small fix in `mysqld_safe` for some shells.
- Give error if a `MyISAM MERGE` table has more than 2 ^ 32 rows and MySQL was not compiled with `-DBIG_TABLES`.
- Fixed some `ORDER BY ... DESC` problems with `InnoDB` tables.

D.3.14 Alterações na distribuição 4.0.4 (29 Sep 2002)

- Fixed bug where `GRANT/REVOKE` failed if hostname was given in non-matching case.
- Don't give warning in `LOAD DATA INFILE` when setting a `timestamp` to a string value of `'0'`.
- Fixed bug in `myisamchk -R` mode.
- Fixed bug that caused `mysqld` to crash on `REVOKE`.

- Fixed bug in `ORDER BY` when there is a constant in the `SELECT` statement.
- One didn't get an error message if `mysqld` couldn't open the privilege tables.
- `SET PASSWORD FOR ...` closed the connection in case of errors (bug from 4.0.3).
- Increased max possible `max_allowed_packet` in `mysqld` to 1 GB.
- Fixed bug when doing a multi-line `INSERT` on a table with an `AUTO_INCREMENT` key which was not in the first part of the key.
- Changed `LOAD DATA INFILE` to not recreate index if the table had rows from before.
- Fixed overrun bug when calling `AES_DECRYPT()` with incorrect arguments.
- `--skip-ssl` can now be used to disable SSL in the MySQL clients, even if one is using other SSL options in an option file or previously on the command line.
- Fixed bug in `MATCH ... AGAINST( ... IN BOOLEAN MODE)` used with `ORDER BY`.
- Added `LOCK TABLES` and `CREATE TEMPORARY TABLES` privilege on the database level. One must run the `mysql_fix_privilege_tables` script on old installations to activate these.
- In `SHOW TABLE ... STATUS`, compressed tables sometimes showed up as `dynamic`.
- `SELECT @@[global|session].var_name` didn't report `global | session` in the result column name.
- Fixed problem in replication that `FLUSH LOGS` in a circular replication setup created an infinite number of binary log files. Now a `rotate-binary-log` command in the binary log will not cause slaves to rotate logs.
- Removed `STOP EVENT` from binary log when doing `FLUSH LOGS`.
- Disable the use of `SHOW NEW MASTER FOR SLAVE` as this needs to be completely reworked in a future release.
- Fixed a bug with constant expression (for example, field of a one-row table, or field from a table, referenced by a `UNIQUE` key) appeared in `ORDER BY` part of `SELECT DISTINCT`.
- `--log-binary=a.b.c` now properly strips off `.b.c`.
- `FLUSH LOGS` removed numerical extension for all future update logs.
- `GRANT ... REQUIRE` didn't store the SSL information in the `mysql.user` table if SSL was not enabled in the server.
- `GRANT ... REQUIRE NONE` can now be used to remove SSL information.
- `AND` is now optional between `REQUIRE` options.
- `REQUIRE` option was not properly saved, which could cause strange output in `SHOW GRANTS`.
- Fixed that `mysqld --help` reports correct values for `--datadir` and `--bind-address`.
- Fixed that one can drop UDFs that didn't exist when `mysqld` was started.
- Fixed core dump problem with `SHOW VARIABLES` on some 64 bit systems (like Solaris sparc).
- Fixed a bug in `my_getopt()`; `--set-variable` syntax didn't work for those options that didn't have a valid variable in the `my_option` struct. This affected at least the `default-table-type` option.
- Fixed a bug from 4.0.2 that caused `REPAIR TABLE` and `myisamchk --recover` to fail on tables with duplicates in a unique key.

- Fixed a bug from 4.0.3 in calculating the default datatype for some functions. This affected queries of type `CREATE TABLE table_name SELECT expression(),...`
- Fixed bug in queries of type `SELECT * FROM table-list GROUP BY ...` and `SELECT DISTINCT * FROM ....`
- Fixed bug with the `--slow-log` when logging an administrator command (like `FLUSH TABLES`).
- Fixed a bug that `OPTIMIZE` of locked and modified table, reported table corruption.
- Fixed a bug in `my_getopt()` in handling of special prefixes (`--skip-`, `--enable-`). `--skip-external-locking` didn't work and the bug may have affected other similar options.
- Fixed bug in checking for output file name of the `tee` option.
- Added some more optimization to use index for `SELECT ... FROM many_tables .. ORDER BY key limit #`
- Fixed problem in `SHOW OPEN TABLES` when a user didn't have access permissions to one of the opened tables.

D.3.15 Alterações na distribuição 4.0.3 (26 Aug 2002: Beta)

- Fixed problem with types of user variables. (Bug #551)
- Fixed problem with `configure ... --localstatedir=...`.
- Cleaned up `mysql.server` script.
- Fixed a bug in `mysqladmin shutdown` when pid file was modified while `mysqladmin` was still waiting for the previous one to disappear. This could happen during a very quick restart and caused `mysqladmin` to hang until `shutdown_timeout` seconds had passed.
- Don't increment warnings when setting `AUTO_INCREMENT` columns to `NULL` in `LOAD DATA INFILE`.
- Fixed all boolean type variables/options to work with the old syntax, for example, all of these work: `--lower-case-table-names`, `--lower-case-table-names=1`, `-O lower-case-table-names=1`, `--set-variable=lower-case-table-names=1`
- Fixed shutdown problem (SIGTERM signal handling) on Solaris. (Bug from 4.0.2).
- `SHOW MASTER STATUS` now returns an empty set if binary log is not enabled.
- `SHOW SLAVE STATUS` now returns an empty set if slave is not initialised.
- Don't update MyISAM index file on update if not strictly necessary.
- Fixed bug in `SELECT DISTINCT ... FROM many_tables ORDER BY not-used-column`.
- Fixed a bug with `BIGINT` values and quoted strings.
- Added `QUOTE()` function that performs SQL quoting to produce values that can be used as data values in queries.
- Changed variable `DELAY_KEY_WRITE` to an enum to allow one set `DELAY_KEY_WRITE` for all tables without taking down the server.
- Changed behaviour of `IF(condition,column,NULL)` so that it returns the value of the column type.

- Made `safe_mysqld` a symlink to `mysqld_safe` in binary distribution.
- Fixed security bug when having an empty database name in the `user.db` table.
- Fixed some problems with `CREATE TABLE ... SELECT function()`.
- `mysqld` now has the option `--temp-pool` enabled by default as this gives better performance with some operating systems.
- Fixed problem with too many allocated alarms on slave when connecting to master many times (normally not a very critical error).
- Fixed hang in `CHANGE MASTER TO` if the slave thread died very quickly.
- Big cleanup in replication code (less logging, better error messages, etc..)
- If the `--code-file` option is specified, the server calls `setrlimit()` to set the maximum allowed core file size to unlimited, so core files can be generated.
- Fixed bug in query cache after temporary table creation.
- Added `--count=N` (`-c`) option to `mysqladmin`, to make the program do only `N` iterations. To be used with `--sleep` (`-i`). Useful in scripts.
- Fixed bug in multi-table `UPDATE`: when updating a table, `do_select()` became confused about reading records from a cache.
- Fixed bug in multi-table `UPDATE` when several fields were referenced from a single table
- Fixed bug in truncating nonexisting table.
- Fixed bug in `REVOKE` that caused user resources to be randomly set.
- Fixed bug in `GRANT` for the new `CREATE TEMPORARY TABLE` privilege.
- Fixed bug in multi-table `DELETE` when tables are re-ordered in the table initialisation method and ref_lengths are of different sizes.
- Fixed two bugs in `SELECT DISTINCT` with large tables.
- Fixed bug in query cache initialisation with very small query cache size.
- Allow `DEFAULT` with `INSERT` statement.
- The startup parameters `myisam_max_sort_file_size` and `myisam_max_extra_sort_file_size` are now given in bytes, not megabytes.
- External system locking of `MyISAM/ISAM` files is now turned off by default. One can turn this on with `--external-locking`. (For most users this is never needed).
- Fixed core dump bug with `INSERT ... SET db_name.table_name.colname=''`.
- Fixed client hangup bug when using some SQL commands with incorrect syntax.
- Fixed a timing bug in `DROP DATABASE`
- New `SET [GLOBAL | SESSION]` syntax to change thread-specific and global server variables at runtime.
- Added variable `slave_compressed_protocol`.
- Renamed variable `query_cache_startup_type` to `query_cache_type`, `myisam_bulk_insert_tree_size` to `bulk_insert_buffer_size`, `record_buffer` to `read_buffer_size` and `record_rnd_buffer` to `read_rnd_buffer_size`.
- Renamed some SQL variables, but old names will still work until 5.0. Veja Seção 2.5.2 [Upgrading-from-3.23], Página 123.
- Renamed `--skip-locking` to `--skip-external-locking`.

- Removed unused variable `query_buffer_size`.
- Fixed a bug that made the pager option in the `mysql` client non-functional.
- Added full `AUTO_INCREMENT` support to `MERGE` tables.
- Extended `LOG()` function to accept an optional arbitrary base parameter. Veja Seção 6.3.3.2 [Mathematical functions], Página 523.
- Added `LOG2()` function (useful for finding out how many bits a number would require for storage).
- Added `LN()` natural logarithm function for compatibility with other databases. It is synonymous with `LOG(X)`.

D.3.16 Alterações na distribuição 4.0.2 (01 Jul 2002)

- Cleaned up `NULL` handling for default values in `DESCRIBE table_name`.
- Fixed `truncate()` to round up negative values to the nearest integer.
- Changed `--chroot=path` option to execute `chroot()` immediately after all options have been parsed.
- Don't allow database names that contain '\'.
- `lower_case_table_names` now also affects database names.
- Added `XOR` operator (logical and bitwise `XOR`) with `^` as a synonym for bitwise `XOR`.
- Added function `IS_FREE_LOCK("lock_name")`. Based on code contributed by Hartmut Holzgraefe `hartmut@six.de`.
- Removed `mysql_ssl_clear()` from C API, as it was not needed.
- `DECIMAL` and `NUMERIC` types can now read exponential numbers.
- Added `SHA1()` function to calculate 160 bit hash value as described in RFC 3174 (Secure Hash Algorithm). This function can be considered a cryptographically more secure equivalent of `MD5()`. Veja Seção 6.3.6.2 [Miscellaneous functions], Página 547.
- Added `AES_ENCRYPT()` and `AES_DECRYPT()` functions to perform encryption according to AES standard (Rijndael). Veja Seção 6.3.6.2 [Miscellaneous functions], Página 547.
- Added `--single-transaction` option to `mysqldump`, allowing a consistent dump of `InnoDB` tables. Veja Seção 4.9.7 [mysqldump], Página 361.
- Fixed bug in `innodb_log_group_home_dir` in `SHOW VARIABLES`.
- Fixed a bug in optimiser with merge tables when non-unique values are used in summing up (causing crashes).
- Fixed a bug in optimiser when a range specified makes index grouping impossible (causing crashes).
- Fixed a rare bug when `FULLTEXT` index is present and no tables are used.
- Added privileges `CREATE TEMPORARY TABLES`, `EXECUTE`, `LOCK TABLES`, `REPLICATION CLIENT`, `REPLICATION SLAVE`, `SHOW DATABASES` and `SUPER`. To use these, you must have run the `mysql_fix_privilege_tables` script after upgrading.
- Fixed query cache align data bug.
- Fixed mutex bug in replication when reading from master fails.

- Added missing mutex in `TRUNCATE TABLE`; This fixes some core dump/hangup problems when using `TRUNCATE TABLE`.
- Fixed bug in multiple-table `DELETE` when optimiser uses only indexes.
- Fixed that `ALTER TABLE table_name RENAME new_table_name` is as fast as `RENAME TABLE`.
- Fixed bug in `GROUP BY` with two or more fields, where at least one field can contain `NULL` values.
- Use `Turbo Boyer-Moore` algorithm to speed up `LIKE "%keyword%"` searches.
- Fixed bug in `DROP DATABASE` with symlink.
- Fixed crash in `REPAIR ... USE_FRM`.
- Fixed bug in `EXPLAIN` with `LIMIT offset != 0`.
- Fixed bug in phrase operator `"..."` in boolean full-text search.
- Fixed bug that caused duplicated rows when using truncation operator `*` in boolean full-text search.
- Fixed bug in truncation operator of boolean full-text search (incorrect results when there are only `+word*`s in the query).
- Fixed bug in boolean full-text search that caused a crash when an identical `MATCH` expression that did not use an index appeared twice.
- Query cache is now automatically disabled in `mysqldump`.
- Fixed problem on Windows 98 that made sending of results very slow.
- Boolean full-text search weighting scheme changed to something more reasonable.
- Fixed bug in boolean full-text search that caused MySQL to ignore queries of `ft_min_word_len` characters.
- Boolean full-text search now supports "phrase searches".
- New configure option `--without-query-cache`.
- Memory allocation strategy for "root memory" changed. Block size now grows with number of allocated blocks.
- `INET_NTOA()` now returns `NULL` if you give it an argument that is too large (greater than the value corresponding to `255.255.255.255`).
- Fix `SQL_CALC_FOUND_ROWS` to work with `UNION`s. It will work only if the first `SELECT` has this option and if there is global `LIMIT` for the entire statement. For the moment, this requires using parentheses for individual `SELECT` queries within the statement.
- Fixed bug in `SQL_CALC_FOUND_ROWS` and `LIMIT`.
- Don't give an error for `CREATE TABLE ...(... VARCHAR(0))`.
- Fixed `SIGINT` and `SIGQUIT` problems in '`mysql.cc`' on Linux with some `glibc` versions.
- Fixed bug in '`convert.cc`', which is caused by having an incorrect `net_store_length()` linked in the `CONVERT::store()` method.
- `DOUBLE` and `FLOAT` columns now honor the `UNSIGNED` flag on storage.
- `InnoDB` now retains foreign key constraints through `ALTER TABLE` and `CREATE/DROP INDEX`.
- `InnoDB` now allows foreign key constraints to be added through the `ALTER TABLE` syntax.

- InnoDB tables can now be set to automatically grow in size (autoextend).
- Added --ignore-lines=n option to mysqlimport. This has the same effect as the IGNORE n LINES clause for LOAD DATA.
- Fixed bug in UNION with last offset being transposed to total result set.
- REPAIR ... USE_FRM added.
- Fixed that DEFAULT_SELECT_LIMIT is always imposed on UNION result set.
- Fixed that some SELECT options can appear only in the first SELECT.
- Fixed bug with LIMIT with UNION, where last select is in the braces.
- Fixed that full-text works fine with UNION operations.
- Fixed bug with indexless boolean full-text search.
- Fixed bug that sometimes appeared when full-text search was used with const tables.
- Fixed incorrect error value when doing a SELECT with an empty HEAP table.
- Use ORDER BY column DESC now sorts NULL values first. (In other words, NULL values sort first in all cases, whether or not DESC is specified.) This is changed back in 4.0.10.
- Fixed bug in WHERE key_name='constant' ORDER BY key_name DESC.
- Fixed bug in SELECT DISTINCT ... ORDER BY DESC optimization.
- Fixed bug in ... HAVING 'GROUP_FUNCTION'(xxx) IS [NOT] NULL.
- Fixed bug in truncation operator for boolean full-text search.
- Allow value of --user=# option for mysqld to be specified as a numeric user ID.
- Fixed a bug where SQL_CALC_ROWS returned an incorrect value when used with one table and ORDER BY and with InnoDB tables.
- Fixed that SELECT 0 LIMIT 0 doesn't hang thread.
- Fixed some problems with USE/IGNORE INDEX when using many keys with the same start column.
- Don't use table scan with BerkeleyDB and InnoDB tables when we can use an index that covers the whole row.
- Optimized InnoDB sort-buffer handling to take less memory.
- Fixed bug in multi-table DELETE and InnoDB tables.
- Fixed problem with TRUNCATE and InnoDB tables that produced the error Can't execute the given command because you have active locked tables or an active transaction.
- Added NO_UNSIGNED_SUBTRACTION to the set of flags that may be specified with the --sql-mode option for mysqld. It disables unsigned arithmetic rules when it comes to subtraction. (This will make MySQL 4.0 behave more like 3.23 with UNSIGNED columns).
- The result returned for all bit functions (|, <<, ...) is now of type unsigned integer.
- Added detection of nan values in MyISAM to make it possible to repair tables with nan in float or double columns.
- Fixed new bug in myisamchk where it didn't correctly update number of "parts" in the MyISAM index file.
- Changed to use autoconf 2.52 (from autoconf 2.13).

- Fixed optimization problem where the MySQL Server was in "preparing" state for a long time when selecting from an empty table which had contained a lot of rows.
- Fixed bug in complicated join with `const` tables. This fix also improves performance a bit when referring to another table from a `const` table.
- First pre-version of multi-table `UPDATE` statement.
- Fixed bug in multi-table `DELETE`.
- Fixed bug in `SELECT CONCAT(argument_list) ... GROUP BY 1`.
- `INSERT ... SELECT` did a full rollback in case of an error. Fixed so that we only roll back the last statement in the current transaction.
- Fixed bug with empty expression for boolean full-text search.
- Fixed core dump bug in updating full-text key from/to `NULL`.
- ODBC compatibility: Added `BIT_LENGTH()` function.
- Fixed core dump bug in `GROUP BY BINARY column`.
- Added support for `NULL` keys in `HEAP` tables.
- Use index for `ORDER BY` in queries of type: `SELECT * FROM t WHERE key_part1=1 ORDER BY key_part1 DESC,key_part2 DESC`
- Fixed bug in `FLUSH QUERY CACHE`.
- Added `CAST()` and `CONVERT()` functions. The `CAST` and `CONVERT` functions are nearly identical and mainly useful when you want to create a column with a specific type in a `CREATE ... SELECT` statement. For more information, read Seção 6.3.5 [Cast Functions], Página 544.
- `CREATE ... SELECT` on `DATE` and `TIME` functions now create columns of the expected type.
- Changed order in which keys are created in tables.
- Added new columns `Null` and `Index_type` to `SHOW INDEX` output.
- Added `--no-beep` and `--prompt` options to `mysql` command-line client.
- New feature: management of user resources.

```
GRANT ... WITH MAX_QUERIES_PER_HOUR N1
               MAX_UPDATES_PER_HOUR N2
               MAX_CONNECTIONS_PER_HOUR N3;
```

  Veja Seção 4.4.7 [User resources], Página 265.
- Added `mysql_secure_installation` to the '`scripts/`' directory.

D.3.17 Alterações na distribuição 4.0.1 (23 Dec 2001)

- Added `system` command to `mysql`.
- Fixed bug when `HANDLER` was used with some unsupported table type.
- `mysqldump` now puts `ALTER TABLE tbl_name DISABLE KEYS` and `ALTER TABLE tbl_name ENABLE KEYS` in the sql dump.
- Added `mysql_fix_extensions` script.
- Fixed stack overrun problem with `LOAD DATA FROM MASTER` on OSF/1.

- Fixed shutdown problem on HP-UX.
- Added `DES_ENCRYPT()` and `DES_DECRYPT()` functions.
- Added `FLUSH DES_KEY_FILE` statement.
- Added `--des-key-file` option to `mysqld`.
- `HEX(string)` now returns the characters in `string` converted to hexadecimal.
- Fixed problem with `GRANT` when using `lower_case_table_names=1`.
- Changed `SELECT ... IN SHARE MODE` to `SELECT ... LOCK IN SHARE MODE` (as in MySQL 3.23).
- A new query cache to cache results from identical `SELECT` queries.
- Fixed core dump bug on 64-bit machines when it got an incorrect communication packet.
- `MATCH ... AGAINST(... IN BOOLEAN MODE)` can now work without `FULLTEXT` index.
- Fixed slave to replicate from 3.23 master.
- Miscellaneous replication fixes/cleanup.
- Got shutdown to work on Mac OS X.
- Added `myisam/ft_dump` utility for low-level inspection of `FULLTEXT` indexes.
- Fixed bug in `DELETE ... WHERE ... MATCH ...`.
- Added support for `MATCH ... AGAINST(... IN BOOLEAN MODE)`. **Note: you must rebuild your tables with `ALTER TABLE tablename TYPE=MyISAM` to be able to use boolean full-text search**.
- `LOCATE()` and `INSTR()` are now case-sensitive if either argument is a binary string.
- Changed `RAND()` initialisation so that `RAND(N)` and `RAND(N+1)` are more distinct.
- Fixed core dump bug in `UPDATE ... ORDER BY`.
- In 3.23, `INSERT INTO ... SELECT` always had `IGNORE` enabled. Now MySQL will stop (and possibly roll back) by default in case of an error unless you specify `IGNORE`.
- Ignore `DATA DIRECTORY` and `INDEX DIRECTORY` directives on Windows.
- Added boolean full-text search code. It should be considered early alpha.
- Extended `MODIFY` and `CHANGE` in `ALTER TABLE` to accept the `FIRST` and `AFTER` keywords.
- Indexes are now used with `ORDER BY` on a whole `InnoDB` table.

D.3.18 Alterações na distribuição 4.0.0 (Oct 2001: Alpha)

- Added `--xml` option to `mysql` for producing XML output.
- Added full-text variables `ft_min_word_len`, `ft_max_word_len`, and `ft_max_word_len_for_sort`.
- Added documentation for `libmysqld`, the embedded MySQL server library. Also added example programs (a `mysql` client and `mysqltest` test program) which use `libmysqld`.
- Removed all Gemini hooks from MySQL server.
- Removed `my_thread_init()` and `my_thread_end()` from '`mysql_com.h`', and added `mysql_thread_init()` and `mysql_thread_end()` to '`mysql.h`'.

- Support for communication packets > 16M. In 4.0.1 we will extend `MyISAM` to be able to handle these.
- Secure connections (with SSL).
- Unsigned `BIGINT` constants now work. `MIN()` and `MAX()` now handle signed and unsigned `BIGINT` numbers correctly.
- New character set `latin1_de` which provides correct German sorting.
- `STRCMP()` now uses the current character set when doing comparisons, which means that the default comparison behaviour now is case-insensitive.
- `TRUNCATE TABLE` and `DELETE FROM tbl_name` are now separate functions. One bonus is that `DELETE FROM tbl_name` now returns the number of deleted rows, rather than zero.
- `DROP DATABASE` now executes a `DROP TABLE` on all tables in the database, which fixes a problem with `InnoDB` tables.
- Added support for `UNION`.
- Added support for multi-table `DELETE` operations.
- A new `HANDLER` interface to `MyISAM` tables.
- Added support for `INSERT` on `MERGE` tables. Patch from Benjamin Pflugmann.
- Changed `WEEK(date,0)` to match the calendar in the USA.
- `COUNT(DISTINCT)` is about 30% faster.
- Speed up all internal list handling.
- Speed up `IS NULL`, `ISNULL()` and some other internal primitives.
- Full-text index creation now is much faster.
- Tree-like cache to speed up bulk inserts and `myisam_bulk_insert_tree_size` variable.
- Searching on packed (`CHAR`/`VARCHAR`) keys is now much faster.
- Optimized queries of type: `SELECT DISTINCT * from tbl_name ORDER by key_part1 LIMIT row_count`.
- `SHOW CREATE TABLE` now shows all table attributes.
- `ORDER BY ... DESC` can now use keys.
- `LOAD DATA FROM MASTER` "automatically" sets up a slave.
- Renamed `safe_mysqld` to `mysqld_safe` to make this name more in line with other MySQL scripts/commands.
- Added support for symbolic links to `MyISAM` tables. Symlink handling is now enabled by default for Windows.
- Added `SQL_CALC_FOUND_ROWS` and `FOUND_ROWS()`. This makes it possible to know how many rows a query would have returned without a `LIMIT` clause.
- Changed output format of `SHOW OPEN TABLES`.
- Allow `SELECT expression LIMIT ...`.
- Added `ORDER BY` syntax to `UPDATE` and `DELETE`.
- `SHOW INDEXES` is now a synonym for `SHOW INDEX`.
- Added `ALTER TABLE tbl_name DISABLE KEYS` and `ALTER TABLE tbl_name ENABLE KEYS` commands.

- Allow use of `IN` as a synonym for `FROM` in `SHOW` commands.
- Implemented "repair by sort" for `FULLTEXT` indexes. `REPAIR TABLE`, `ALTER TABLE`, and `OPTIMIZE TABLE` for tables with `FULLTEXT` indexes are now up to 100 times faster.
- Allow SQL-99 syntax `X'hexadecimal-number'`.
- Cleaned up global lock handling for `FLUSH TABLES WITH READ LOCK`.
- Fixed problem with `DATETIME = constant` in `WHERE` optimization.
- Added `--master-data` and `--no-autocommit` options to `mysqldump`. (Thanks to Brian Aker for this.)
- Added script `mysql_explain_log.sh` to distribution. (Thanks to mobile.de).

D.4 Alterações na distribuição 3.23.x (Recent; still supported)

Please note that since release 4.0 is now production level, only critical fixes are done in the 3.23 release series. You are recommended to upgrade when possible, to take advantage of all speed and feature improvements in 4.0. Veja Seção 2.5.2 [Upgrading-from-3.23], Página 123.

The 3.23 release has several major features that are not present in previous versions. We have added three new table types:

`MyISAM` A new ISAM library which is tuned for SQL and supports large files.

`InnoDB` A transaction-safe storage engine that supports row level locking, and many Oracle-like features.

`BerkeleyDB` or `BDB`
Uses the Berkeley DB library from Sleepycat Software to implement transaction-safe tables.

Note that only `MyISAM` is available in the standard binary distribution.

The 3.23 release also includes support for database replication between a master and many slaves, full-text indexing, and much more.

All new features are being developed in the 4.x version. Only bug fixes and minor enhancements to existing features will be added to 3.23.

The replication code and BerkeleyDB code is still not as tested and as the rest of the code, so we will probably need to do a couple of future releases of 3.23 with small fixes for this part of the code. As long as you don't use these features, you should be quite safe with MySQL 3.23!

Note that the above doesn't mean that replication or Berkeley DB don't work. We have done a lot of testing of all code, including replication and `BDB` without finding any problems. It only means that not as many users use this code as the rest of the code and because of this we are not yet 100% confident in this code.

D.4.1 Alterações na distribuição 3.23.59 (not released yet)

- If a query was ignored on the slave (because of `replicate-ignore-table` and other similar rules), the slave still checked if the query got the same error code (0, no error)

as on the master. So if the master had an error on the query (for example, "Duplicate entry" in a multiple-row insert), then the slave stopped and warned that the error codes didn't match. This is a backport of the fix for MySQL 4.0. (Bug #797)

- `mysqlbinlog` now asks for a password at console when the `-p/--password` option is used with no argument. This is how the other clients (`mysqladmin`, `mysqldump`..) already behave. Note that one now has to use `mysqlbinlog -p<my_password>`; `mysqlbinlog -p <my_password>` will not work anymore (in other words, put no space after `-p`). (Bug #1595)
- On some 64-bit machines (some HP-UX and Solaris machines), a slave installed with the 64-bit MySQL binary could not connect to its master (it connected to itself instead). (Bug #1256, #1381)
- Fixed a Windows-specific bug present since MySQL version 3.23.57 and 3.23.58, which caused Windows slaves to crash when they started replication if a '`master.info`' file existed. (Bug #1720)

D.4.2 Alterações na distribuição 3.23.58 (11 Sep 2003)

- Fixed buffer overflow in password handling which could potentially be exploited by MySQL users with `ALTER` privilege on the `mysql.user` table to execute random code or to gain shell access with the UID of the mysqld process (thanks to Jedi/Sector One for spotting and reporting this bug).
- `mysqldump` now correctly quotes all identifiers when communicating with the server. This assures that during the dump process, `mysqldump` will never send queries to the server that result in a syntax error. This problem is **not** related to the `mysqldump` program's output, which was not changed. (Bug #1148)
- Fixed table/column grant handling - proper sort order (from most specific to less specific, veja Seção 4.3.10 [Request access], Página 242) was not honored. (Bug #928)
- Fixed overflow bug in `MyISAM` and `ISAM` when a row is updated in a table with a large number of columns and at least one `BLOB/TEXT` column.
- Fixed MySQL so that field length (in C API) for the second column in `SHOW CREATE TABLE` is always larger than the data length. The only known application that was affected by the old behaviour was Borland dbExpress, which truncated the output from the command. (Bug #1064)
- Fixed `ISAM` bug in `MAX()` optimization.
- Fixed `Unknown error` when doing `ORDER BY` on reference table which was used with `NULL` value on `NOT NULL` column. (Bug #479)

D.4.3 Alterações na distribuição 3.23.57 (06 Jun 2003)

- Fixed problem in alarm handling that could cause problems when getting a packet that is too large.
- Fixed problem when installing MySQL as a service on Windows when one gave 2 arguments (option file group name and service name) to `mysqld`.
- Fixed `kill pid-of-mysqld` to work on Mac OS X.

- `SHOW TABLE STATUS` displayed incorrect `Row_format` value for tables that have been compressed with `myisampack`. (Bug #427)
- `SHOW VARIABLES LIKE 'innodb_data_file_path'` displayed only the name of the first datafile. (Bug #468)
- Fixed security problem where `mysqld` didn't allow one to `UPDATE` rows in a table even if one had a global `UPDATE` privilege and a database `SELECT` privilege.
- Fixed a security problem with `SELECT` and wildcarded select list, when user only had partial column `SELECT` privileges on the table.
- Fixed unlikely problem in optimising `WHERE` clause with a constant expression such as in `WHERE 1 AND (a=1 AND b=1)`.
- Fixed problem on IA-64 with timestamps that caused `mysqlbinlog` to fail.
- The default option for `innodb_flush_log_at_trx_commit` was changed from 0 to 1 to make `InnoDB` tables ACID by default. Veja Seção 7.5.3 [InnoDB start], Página 644.
- Fixed problem with too many allocated alarms on slave when connecting to master many times (normally not a very critical error).
- Fixed a bug in replication of temporary tables. (Bug #183)
- Fixed 64 bit bug that affected at least AMD hammer systems.
- Fixed a bug when doing `LOAD DATA INFILE IGNORE`: When reading the binary log, `mysqlbinlog` and the replication code read `REPLACE` instead of `IGNORE`. This could make the slave's table become different from the master's table. (Bug #218)
- Fixed overflow bug in `MyISAM` when a row is inserted into a table with a large number of columns and at least one `BLOB/TEXT` column. Bug was caused by incorrect calculation of the needed buffer to pack data.
- The binary log was not locked during `TRUNCATE table_name` or `DELETE FROM table_name` statements, which could cause an `INSERT` to `table_name` to be written to the log before the `TRUNCATE` or `DELETE` statements.
- Fixed rare bug in `UPDATE` of `InnoDB` tables where one row could be updated multiple times.
- Produce an error for empty table and column names.
- Changed `PROCEDURE ANALYSE()` to report `DATE` instead of `NEWDATE`.
- Changed `PROCEDURE ANALYSE(#)` to restrict the number of values in an `ENUM` column to `#` also for string values.
- `mysqldump` no longer silently deletes the binary logs when invoked with the `--master-data` or `--first-slave` option; while this behaviour was convenient for some users, others may suffer from it. Now one has to explicitly ask for binary logs to be deleted by using the new `--delete-master-logs` option.
- Fixed a bug in `mysqldump` when it was invoked with the `--master-data` option: The `CHANGE MASTER TO` statements that were appended to the SQL dump had incorrect coordinates. (Bug #159)

D.4.4 Alterações na distribuição 3.23.56 (13 Mar 2003)

- Fixed `mysqld` crash on extremely small values of `sort_buffer` variable.

- Fixed a bug in privilege system for `GRANT UPDATE` on column level.
- Fixed a rare bug when using a date in `HAVING` with `GROUP BY`.
- Fixed checking of random part of `WHERE` clause. (Bug #142)
- Fixed MySQL (and `myisamchk`) crash on artificially corrupted '`.MYI`' files.
- Security enhancement: `mysqld` no longer reads options from world-writeable config files.
- Security enhancement: `mysqld` and `safe_mysqld` now only use the first `--user` option specified on the command line. (Normally this comes from '`/etc/my.cnf`')
- Security enhancement: Don't allow `BACKUP TABLE` to overwrite existing files.
- Fixed unlikely deadlock bug when one thread did a `LOCK TABLE` and another thread did a `DROP TABLE`. In this case one could do a `KILL` on one of the threads to resolve the deadlock.
- `LOAD DATA INFILE` was not replicated by slave if `replicate_*_table` was set on the slave.
- Fixed a bug in handling `CHAR(0)` columns that could cause incorrect results from the query.
- Fixed a bug in `SHOW VARIABLES` on 64-bit platforms. The bug was caused by incorrect declaration of variable `server_id`.
- The Comment column in `SHOW TABLE STATUS` now reports that it can contain `NULL` values (which is the case for a crashed '`.frm`' file).
- Fixed the `rpl_rotate_logs` test to not fail on certain platforms (e.g. Mac OS X) due to a too long file name (changed `slave-master-info.opt` to `.slave-mi`).
- Fixed a problem with `BLOB NOT NULL` columns used with `IS NULL`.
- Fixed bug in `MAX()` optimization in `MERGE` tables.
- Better `RAND()` initialisation for new connections.
- Fixed bug with connect timeout. This bug was manifested on OS's with `poll()` system call, which resulted in timeout the value specified as it was executed in both `select()` and `poll()`.
- Fixed bug in `SELECT * FROM table WHERE datetime1 IS NULL OR datetime2 IS NULL`.
- Fixed bug in using aggregate functions as argument for `INTERVAL`, `CASE`, `FIELD`, `CONCAT_WS`, `ELT` and `MAKE_SET` functions.
- When running with `--lower-case-table-names` (default on Windows) and you had tables or databases with mixed case on disk, then executing `SHOW TABLE STATUS` followed with `DROP DATABASE` or `DROP TABLE` could fail with `Errcode 13`.
- Fixed bug in logging to binary log (which affects replication) a query that inserts a `NULL` in an `auto_increment` field and also uses `LAST_INSERT_ID()`.
- Fixed bug in `mysqladmin --relative`.
- On some 64 bit systems, `show status` reported a strange number for `Open_files` and `Open_streams`.

D.4.5 Alterações na distribuição 3.23.55 (23 Jan 2003)

- Fixed double `free`'d pointer bug in `mysql_change_user()` handling, that enabled a specially hacked version of MySQL client to crash `mysqld`. **Note**, that one needs to login to the server by using a valid user account to be able to exploit this bug.
- Fixed bug with the `--slow-log` when logging an administrator command (like `FLUSH TABLES`).
- Fixed bug in `GROUP BY` when used on BLOB column with `NULL` values.
- Fixed a bug in handling `NULL` values in `CASE ... WHEN ...`.
- Bugfix for `--chroot` (veja Seção D.4.6 [News-3.23.54], Página 1004) is reverted. Unfortunately, there is no way to make it to work, without introducing backward-incompatible changes in '`my.cnf`'. Those who need `--chroot` functionality, should upgrade to MySQL 4.0. (The fix in the 4.0 branch did not break backward-compatibility).
- Make `--lower-case-table-names` default on Mac OS X as the default file system (HFS+) is case insensitive.
- Fixed a bug in '`scripts/mysqld_safe.sh`' in `NOHUP_NICENESS` testing.
- Transactions in `AUTOCOMMIT=0` mode didn't rotate binary log.
- Fixed a bug in `scripts/make_binary_distribution` that resulted in a remaining `@HOSTNAME@` variable instead of replacing it with the correct path to the `hostname` binary.
- Fixed a very unlikely bug that could cause `SHOW PROCESSLIST` to core dump in pthread_mutex_unlock() if a new thread was connecting.
- Forbid `SLAVE STOP` if the thread executing the query has locked tables. This removes a possible deadlock situation.

D.4.6 Alterações na distribuição 3.23.54 (05 Dec 2002)

- Fixed a bug, that allowed to crash `mysqld` with a specially crafted packet.
- Fixed a rare crash (double `free`'d pointer) when altering a temporary table.
- Fixed buffer overrun in `libmysqlclient` library that allowed malicious MySQL server to crash the client application.
- Fixed security-related bug in `mysql_change_user()` handling. All users are strongly recommended to upgrade to the version 3.23.54.
- Fixed bug that prevented `--chroot` command-line option of `mysqld` from working.
- Fixed bug that made `OPTIMIZE TABLE` to corrupt the table under some rare circumstances.
- Fixed `mysqlcheck` so it can deal with table names containing dashes.
- Fixed shutdown problem on Mac OS X.
- Fixed bug with comparing an indexed `NULL` field with `<=> NULL`.
- Fixed bug that caused `IGNORE INDEX` and `USE INDEX` sometimes to be ignored.
- Fixed rare core dump problem in complicated `GROUP BY` queries that didn't return any result.

- Fixed a bug where `MATCH ... AGAINST () >=0` was treated as if it was `>`.
- Fixed core dump in `SHOW PROCESSLIST` when running with an active slave (unlikely timing bug).
- Make it possible to use multiple MySQL servers on Windows (code backported from 4.0.2).
- One can create `TEMPORARY MERGE` tables now.
- Fixed that `--core-file` works on Linux (at least on kernel 2.4.18).
- Fixed a problem with `BDB` and `ALTER TABLE`.
- Fixed reference to freed memory when doing complicated `GROUP BY ... ORDER BY` queries. Symptom was that `mysqld` died in function `send_fields`.
- Allocate heap rows in smaller blocks to get better memory usage.
- Fixed memory allocation bug when storing `BLOB` values in internal temporary tables used for some (unlikely) `GROUP BY` queries.
- Fixed a bug in key optimising handling where the expression `WHERE column_name = key_column_name` was calculated as true for `NULL` values.
- Fixed core dump bug when doing `LEFT JOIN ... WHERE key_column=NULL`.
- Fixed `MyISAM` crash when using dynamic-row tables with huge numbers of packed fields.
- Updated source tree to be built using `automake 1.5` and `libtool 1.4`.

D.4.7 Alterações na distribuição 3.23.53 (09 Oct 2002)

- Fixed crash when `SHOW INNODB STATUS` was used and `skip-innodb` was defined.
- Fixed possible memory corruption bug in binary log file handling when slave rotated the logs (only affected 3.23, not 4.0).
- Fixed problem in `LOCK TABLES` on Windows when one connects to a database that contains upper case letters.
- Fixed that `--skip-show-databases` doesn't reset the `--port` option.
- Small fix in `safe_mysqld` for some shells.
- Fixed that `FLUSH STATUS` doesn't reset `delayed_insert_threads`.
- Fixed core dump bug when using the `BINARY` cast on a `NULL` value.
- Fixed race condition when someone did a `GRANT` at the same time a new user logged in or did a `USE database`.
- Fixed bug in `ALTER TABLE` and `RENAME TABLE` when running with `-O lower_case_table_names=1` (typically on Windows) when giving the table name in uppercase.
- Fixed that `-O lower_case_table_names=1` also converts database names to lower case.
- Fixed unlikely core dump with `SELECT ... ORDER BY ... LIMIT`.
- Changed `AND/OR` to report that they can return NULL. This fixes a bug in `GROUP BY` on `AND/OR` expressions that return `NULL`.
- Fixed a bug that `OPTIMIZE` of locked and modified MyISAM table, reported table corruption.
- Fixed a `BDB`-related `ALTER TABLE` bug with dropping a column and shutting down immediately thereafter.

- Fixed problem with `configure ... --localstatedir=...`.
- Fixed problem with `UNSIGNED BIGINT` on AIX (again).
- Fixed bug in pthread_mutex_trylock() on HPUX 11.0.
- Multi-threaded stress tests for `InnoDB`.

D.4.8 Alterações na distribuição 3.23.52 (14 Aug 2002)

- Wrap `BEGIN/COMMIT` around transaction in the binary log. This makes replication honour transactions.
- Fixed security bug when having an empty database name in the `user.db` table.
- Changed initialisation of `RND()` to make it less predicatable.
- Fixed problem with `GROUP BY` on result with expression that created a `BLOB` field.
- Fixed problem with `GROUP BY` on columns that have `NULL` values. To solve this we now create an `MyISAM` temporary table when doing a `GROUP BY` on a possible `NULL` item. From MySQL 4.0.5 we can use in memory `HEAP` tables for this case.
- Fixed problem with privilege tables when downgrading from 4.0.2 to 3.23.
- Fixed thread bug in `SLAVE START`, `SLAVE STOP` and automatic repair of MyISAM tables that could cause table cache to be corrupted.
- Fixed possible thread related key-cache-corruption problem with `OPTIMIZE TABLE` and `REPAIR TABLE`.
- Added name of 'administrator command' logs.
- Fixed bug with creating an auto-increment value on second part of a `UNIQUE()` key where first part could contain `NULL` values.
- Don't write slave-timeout reconnects to the error log.
- Fixed bug with slave net read timeouting
- Fixed a core-dump bug with `MERGE` tables and `MAX()` function.
- Fixed bug in `ALTER TABLE` with `BDB` tables.
- Fixed bug when logging `LOAD DATA INFILE` to binary log with no active database.
- Fixed a bug in range optimiser (causing crashes).
- Fixed possible problem in replication when doing `DROP DATABASE` on a database with `InnoDB` tables.
- Fixed that `mysql_info()` returns 0 for 'Duplicates' when using `INSERT DELAYED IGNORE`.
- Added `-DHAVE_BROKEN_REALPATH` to the Mac OS X (darwin) compile options in '`configure.in`' to fix a failure under high load.

D.4.9 Alterações na distribuição 3.23.51 (31 May 2002)

- Fix bug with closing tags missing slash for `mysqldump` XML output.
- Remove end space from `ENUM` values. (This fixed a problem with `SHOW CREATE TABLE`.)
- Fixed bug in `CONCAT_WS()` that cut the result.

- Changed name of server variables `Com_show_master_stat` to `Com_show_master_status` and `Com_show_slave_stat` to `Com_show_slave_status`.
- Changed handling of `gethostbyname()` to make the client library thread-safe even if `gethostbyname_r` doesn't exist.
- Fixed core-dump problem when giving a wrong password string to `GRANT`.
- Fixed bug in `DROP DATABASE` with symlinked directory.
- Fixed optimization problem with `DATETIME` and value outside `DATETIME` range.
- Removed Sleepycat's `BDB` doc files from the source tree, as they're not needed (MySQL covers `BDB` in its own documentation).
- Fixed MIT-pthreads to compile with `glibc` 2.2 (needed for `make dist`).
- Fixed the `FLOAT(X+1,X)` is not converted to `FLOAT(X+2,X)`. (This also affected `DECIMAL`, `DOUBLE` and `REAL` types)
- Fixed the result from `IF()` is case in-sensitive if the second and third arguments are case sensitive.
- Fixed core dump problem on OSF/1 in `gethostbyname_r`.
- Fixed that underflowed decimal fields are not zero filled.
- If we get an overflow when inserting `'+11111'` for `DECIMAL(5,0) UNSIGNED` columns, we will just drop the sign.
- Fixed optimization bug with `ISNULL(expression_which_cannot_be_null)` and `ISNULL(constant_expression)`.
- Fixed host lookup bug in the `glibc` library that we used with the 3.23.50 Linux-x86 binaries.

D.4.10 Alterações na distribuição 3.23.50 (21 Apr 2002)

- Fixed buffer overflow problem if someone specified a too long datadir parameter to mysqld
- Add missing `<row>` tags for `mysqldump` XML output.
- Fixed problem with `crash-me` and `gcc` 3.0.4.
- Fixed that `@@unknown_variable` doesn't hang server.
- Added `@@VERSION` as a synonym for `VERSION()`.
- `SHOW VARIABLES LIKE 'xxx'` is now case-insensitive.
- Fixed timeout for `GET_LOCK()` on HP-UX with DCE threads.
- Fixed memory allocation bug in the glibc library used to build Linux binaries, which caused mysqld to die in 'free()'.
- Fixed `SIGINT` and `SIGQUIT` problems in `mysql`.
- Fixed bug in character table converts when used with big (> 64K) strings.
- `InnoDB` now retains foreign key constraints through `ALTER TABLE` and `CREATE/DROP INDEX`.
- `InnoDB` now allows foreign key constraints to be added through the `ALTER TABLE` syntax.
- `InnoDB` tables can now be set to automatically grow in size (autoextend).

- Our Linux RPMS and binaries are now compiled with `gcc` 3.0.4, which should make them a bit faster.
- Fixed some buffer overflow problems when reading startup parameters.
- Because of problems on shutdown we have now disabled named pipes on Windows by default. One can enable named pipes by starting mysqld with `--enable-named-pipe`.
- Fixed bug when using `WHERE key_column = 'J' or key_column='j'`.
- Fixed core-dump bug when using `--log-bin` with `LOAD DATA INFILE` without an active database.
- Fixed bug in `RENAME TABLE` when used with `lower_case_table_names=1` (default on Windows).
- Fixed unlikely core-dump bug when using `DROP TABLE` on a table that was in use by a thread that also used queries on only temporary tables.
- Fixed problem with `SHOW CREATE TABLE` and `PRIMARY KEY` when using 32 indexes.
- Fixed that one can use `SET PASSWORD` for the anonymous user.
- Fixed core dump bug when reading client groups from option files using `mysql_options()`.
- Memory leak (16 bytes per every **corrupted** table) closed.
- Fixed binary builds to use `--enable-local-infile`.
- Update source to work with new version of `bison`.
- Updated shell scripts to now agree with new POSIX standard.
- Fixed bug where `DATE_FORMAT()` returned empty string when used with `GROUP BY`.

D.4.11 Alterações na distribuição 3.23.49

- Don't give warning for a statement that is only a comment; this is needed for `mysqldump --disable-keys` to work.
- Fixed unlikely caching bug when doing a join without keys. In this case the last used field for a table always returned `NULL`.
- Added options to make `LOAD DATA LOCAL INFILE` more secure.
- MySQL binary release 3.23.48 for Linux contained a new `glibc` library, which has serious problems under high load and Red Hat 7.2. The 3.23.49 binary release doesn't have this problem.
- Fixed shutdown problem on NT.

D.4.12 Alterações na distribuição 3.23.48 (07 Feb 2002)

- Added `--xml` option to `mysqldump` for producing XML output.
- Changed to use `autoconf` 2.52 (from `autoconf` 2.13)
- Fixed bug in complicated join with `const` tables.
- Added internal safety checks for `InnoDB`.
- Some `InnoDB` variables were always shown in `SHOW VARIABLES` as `OFF` on high-byte-first systems (like SPARC).

- Fixed problem with one thread using an `InnoDB` table and another thread doing an `ALTER TABLE` on the same table. Before that, `mysqld` could crash with an assertion failure in '`row0row.c`', line 474.
- Tuned the `InnoDB` SQL optimiser to favor index searches more often over table scans.
- Fixed a performance problem with `InnoDB` tables when several large `SELECT` queries are run concurrently on a multiprocessor Linux computer. Large CPU-bound `SELECT` queries will now also generally run faster on all platforms.
- If MySQL binlogging is used, `InnoDB` now prints after crash recovery the latest MySQL binlog name and the offset `InnoDB` was able to recover to. This is useful, for example, when resynchronising a master and a slave database in replication.
- Added better error messages to help in installation problems of `InnoDB` tables.
- It is now possible to recover MySQL temporary tables that have become orphaned inside the `InnoDB` tablespace.
- `InnoDB` now prevents a `FOREIGN KEY` declaration where the signedness is not the same in the referencing and referenced integer columns.
- Calling `SHOW CREATE TABLE` or `SHOW TABLE STATUS` could cause memory corruption and make `mysqld` crash. Especially at risk was `mysqldump`, because it frequently calls `SHOW CREATE TABLE`.
- If inserts to several tables containing an `AUTO_INCREMENT` column were wrapped inside one `LOCK TABLES`, `InnoDB` asserted in '`lock0lock.c`'.
- In 3.23.47 we allowed several `NULL` values in a `UNIQUE` secondary index for an `InnoDB` table. But `CHECK TABLE` was not relaxed: it reports the table as corrupt. `CHECK TABLE` no longer complains in this situation.
- `SHOW GRANTS` now shows `REFERENCES` instead of `REFERENCE`.

D.4.13 Alterações na distribuição 3.23.47 (27 Dec 2001)

- Fixed bug when using the following construct: `SELECT ... WHERE key=@var_name OR key=@var_name2`
- Restrict `InnoDB` keys to 500 bytes.
- `InnoDB` now supports `NULL` in keys.
- Fixed shutdown problem on HP-UX. (Introduced in 3.23.46)
- Fixed core dump bug in replication when using `SELECT RELEASE_LOCK()`.
- Added new command: `DO expression,[expression]`
- Added `slave-skip-errors` option.
- Added statistics variables for all MySQL commands. (`SHOW STATUS` is now much longer.)
- Fixed default values for `InnoDB` tables.
- Fixed that `GROUP BY expr DESC` works.
- Fixed bug when using `t1 LEFT JOIN t2 ON t2.key=constant`.
- `mysql_config` now also works with binary (relocated) distributions.

D.4.14 Alterações na distribuição 3.23.46 (29 Nov 2001)

- Fixed problem with aliased temporary table replication.
- `InnoDB` and `BDB` tables will now use index when doing an `ORDER BY` on the whole table.
- Fixed bug where one got an empty set instead of a DEADLOCK error when using `BDB` tables.
- One can now kill `ANALYZE`, `REPAIR`, and `OPTIMIZE TABLE` when the thread is waiting to get a lock on the table.
- Fixed race condition in `ANALYZE TABLE`.
- Fixed bug when joining with caching (unlikely to happen).
- Fixed race condition when using the binary log and `INSERT DELAYED` which could cause the binary log to have rows that were not yet written to `MyISAM` tables.
- Changed caching of binary log to make replication slightly faster.
- Fixed bug in replication on Mac OS X.

D.4.15 Alterações na distribuição 3.23.45 (22 Nov 2001)

- `(UPDATE|DELETE) ...WHERE MATCH` bugfix.
- shutdown should now work on Darwin (Mac OS X).
- Fixed core dump when repairing corrupted packed `MyISAM` files.
- `--core-file` now works on Solaris.
- Fix a bug which could cause `InnoDB` to complain if it cannot find free blocks from the buffer cache during recovery.
- Fixed bug in `InnoDB` insert buffer B-tree handling that could cause crashes.
- Fixed bug in `InnoDB` lock timeout handling.
- Fixed core dump bug in `ALTER TABLE` on a `TEMPORARY InnoDB` table.
- Fixed bug in `OPTIMIZE TABLE` that reset index cardinality if it was up to date.
- Fixed problem with `t1 LEFT_JOIN t2 ... WHERE t2.date_column IS NULL` when date_column was declared as `NOT NULL`.
- Fixed bug with `BDB` tables and keys on `BLOB` columns.
- Fixed bug in `MERGE` tables on OS with 32-bit file pointers.
- Fixed bug in `TIME_TO_SEC()` when using negative values.

D.4.16 Alterações na distribuição 3.23.44 (31 Oct 2001)

- Fixed `Rows_examined` count in slow query log.
- Fixed bug when using a reference to an `AVG()` column in `HAVING`.
- Fixed that date functions that require correct dates, like `DAYOFYEAR(column)`, will return `NULL` for `0000-00-00` dates.
- Fixed bug in const-propagation when comparing columns of different types. (`SELECT * FROM date_col="2001-01-01" and date_col=time_col`)

- Fixed bug that caused error message `Can't write, because of unique constraint` with some `GROUP BY` queries.
- Fixed problem with `sjis` character strings used within quoted table names.
- Fixed core dump when using `CREATE ... FULLTEXT` keys with other storage engines than `MyISAM`.
- Don't use `signal()` on Windows because this appears to not be 100% reliable.
- Fixed bug when doing `WHERE col_name=NULL` on an indexed column that had `NULL` values.
- Fixed bug when doing `LEFT JOIN ... ON (col_name = constant) WHERE col_name = constant`.
- When using replications, aborted queries that contained `%` could cause a core dump.
- `TCP_NODELAY` was not used on some systems. (Speed problem.)
- Applied portability fixes for OS/2. (Patch by Yuri Dario.)

The following changes are for `InnoDB` tables:

- Add missing `InnoDB` variables to `SHOW VARIABLES`.
- Foreign keys checking is now done for `InnoDB` tables.
- `DROP DATABASE` now works also for `InnoDB` tables.
- `InnoDB` now supports datafiles and raw disk partitions bigger than 4 GB on those operating systems that have big files.
- `InnoDB` calculates better table cardinality estimates for the MySQL optimiser.
- Accent characters in the default character set `latin1` are ordered according to the MySQL ordering.

  Note: if you are using `latin1` and have inserted characters whose code is greater than 127 into an indexed `CHAR` column, you should run `CHECK TABLE` on your table when you upgrade to 3.23.44, and drop and reimport the table if `CHECK TABLE` reports an error!
- A new '`my.cnf`' parameter, `innodb_thread_concurrency`, helps in performance tuning in heavily concurrent environments.
- A new '`my.cnf`' parameter, `innodb_fast_shutdown`, speeds up server shutdown.
- A new '`my.cnf`' parameter, `innodb_force_recovery`, helps to save your data in case the disk image of the database becomes corrupt.
- `innodb_monitor` has been improved and a new `innodb_table_monitor` added.
- Increased maximum key length from 500 to 7000 bytes.
- Fixed a bug in replication of `AUTO_INCREMENT` columns with multiple-line inserts.
- Fixed a bug when the case of letters changes in an update of an indexed secondary column.
- Fixed a hang when there are > 24 datafiles.
- Fixed a crash when `MAX(col)` is selected from an empty table, and `col` is not the first column in a multi-column index.
- Fixed a bug in purge which could cause crashes.

D.4.17 Alterações na distribuição 3.23.43 (04 Oct 2001)

- Fixed a bug in `INSERT DELAYED` and `FLUSH TABLES` introduced in 3.23.42.
- Fixed unlikely bug, which returned non-matching rows, in `SELECT` with many tables and multi-column indexes and 'range' type.
- Fixed an unlikely core dump bug when doing `EXPLAIN SELECT` when using many tables and `ORDER BY`.
- Fixed bug in `LOAD DATA FROM MASTER` when using table with `CHECKSUM=1`.
- Added unique error message when one gets a DEADLOCK during a transaction with `BDB` tables.
- Fixed problem with `BDB` tables and `UNIQUE` columns defined as `NULL`.
- Fixed problem with `myisampack` when using pre-space filled `CHAR` columns.
- Applied patch from Yuri Dario for OS/2.
- Fixed bug in `--safe-user-create`.

D.4.18 Alterações na distribuição 3.23.42 (08 Sep 2001)

- Fixed problem when using `LOCK TABLES` and `BDB` tables.
- Fixed problem with `REPAIR TABLE` on `MyISAM` tables with row lengths in the range from 65517 to 65520 bytes.
- Fixed rare hang when doing `mysqladmin shutdown` when there was a lot of activity in other threads.
- Fixed problem with `INSERT DELAYED` where delay thread could be hanging on `upgrading locks` with no apparent reason.
- Fixed problem with `myisampack` and `BLOB`.
- Fixed problem when one edited '`.MRG`' tables by hand. (Patch from Benjamin Pflugmann).
- Enforce that all tables in a `MERGE` table come from the same database.
- Fixed bug with `LOAD DATA INFILE` and transactional tables.
- Fix bug when using `INSERT DELAYED` with wrong column definition.
- Fixed core dump during `REPAIR` of some particularly broken tables.
- Fixed bug in `InnoDB` and `AUTO_INCREMENT` columns.
- Fixed bug in `InnoDB` and `RENAME TABLE` columns.
- Fixed critical bug in `InnoDB` and `BLOB` columns. If you have used `BLOB` columns larger than 8000 bytes in an `InnoDB` table, it is necessary to dump the table with `mysqldump`, drop it and restore it from the dump.
- Applied large patch for OS/2 from Yuri Dario.
- Fixed problem with `InnoDB` when one could get the error `Can't execute the given command...` even when no transaction was active.
- Applied some minor fixes that concern Gemini.
- Use real arithmetic operations even in integer context if not all arguments are integers. (Fixes uncommon bug in some integer contexts).

- Don't force everything to lowercase on Windows. (To fix problem with Windows and `ALTER TABLE`). Now `--lower_case_names` also works on Unix.
- Fixed that automatic rollback is done when thread end doesn't lock other threads.

D.4.19 Alterações na distribuição 3.23.41 (11 Aug 2001)

- Added `--sql-mode=value[,value[,value]]` option to `mysqld`. Veja Seção 4.1.1 [Command-line options], Página 207.
- Fixed possible problem with `shutdown` on Solaris where the '`.pid`' file wasn't deleted.
- `InnoDB` now supports < 4 GB rows. The former limit was 8000 bytes.
- The `doublewrite` file flush method is used in `InnoDB`. It reduces the need for Unix `fsync()` calls to a fraction and improves performance on most Unix flavors.
- You can now use the `InnoDB` Monitor to print a lot of `InnoDB` state information, including locks, to the standard output. This is useful in performance tuning.
- Several bugs which could cause hangs in `InnoDB` have been fixed.
- Split `record_buffer` to `record_buffer` and `record_rnd_buffer`. To make things compatible to previous MySQL versions, if `record_rnd_buffer` is not set, then it takes the value of `record_buffer`.
- Fixed optimising bug in `ORDER BY` where some `ORDER BY` parts where wrongly removed.
- Fixed overflow bug with `ALTER TABLE` and `MERGE` tables.
- Added prototypes for `my_thread_init()` and `my_thread_end()` to '`mysql_com.h`'
- Added `--safe-user-create` option to `mysqld`.
- Fixed bug in `SELECT DISTINCT ... HAVING` that caused error message `Can't find record in #...`

D.4.20 Alterações na distribuição 3.23.40

- Fixed problem with `--low-priority-updates` and `INSERT` statements.
- Fixed bug in slave thread when under some rare circumstances it could get 22 bytes ahead on the offset in the master.
- Added `slave_net_timeout` for replication.
- Fixed problem with `UPDATE` and `BDB` tables.
- Fixed hard bug in `BDB` tables when using key parts.
- Fixed problem when using `GRANT FILE ON database.* ...`; previously we added the `DROP` privilege for the database.
- Fixed `DELETE FROM tbl_name ... LIMIT 0` and `UPDATE FROM tbl_name ... LIMIT 0`, which acted as though the `LIMIT` clause was not present (they deleted or updated all selected rows).
- `CHECK TABLE` now checks if an `AUTO_INCREMENT` column contains the value 0.
- Sending a `SIGHUP` to `mysqld` will now only flush the logs, not reset the replication.
- Fixed parser to allow floats of type `1.0e1` (no sign after `e`).
- Option `--force` to `myisamchk` now also updates states.

- Added option `--warnings` to `mysqld`. Now `mysqld` prints the error `Aborted connection` only if this option is used.
- Fixed problem with `SHOW CREATE TABLE` when you didn't have a `PRIMARY KEY`.
- Properly fixed the rename of `innodb_unix_file_flush_method` variable to `innodb_flush_method`.
- Fixed bug when converting `BIGINT UNSIGNED` to `DOUBLE`. This caused a problem when doing comparisons with `BIGINT` values outside of the signed range.
- Fixed bug in `BDB` tables when querying empty tables.
- Fixed a bug when using `COUNT(DISTINCT)` with `LEFT JOIN` and there weren't any matching rows.
- Removed all documentation referring to the `GEMINI` table type. `GEMINI` is not released under an `Open Source` license.

D.4.21 Alterações na distribuição 3.23.39 (12 Jun 2001)

- The `AUTO_INCREMENT` sequence wasn't reset when dropping and adding an `AUTO_INCREMENT` column.
- `CREATE ... SELECT` now creates non-unique indexes delayed.
- Fixed problem where `LOCK TABLES tbl_name READ` followed by `FLUSH TABLES` put an exclusive lock on the table.
- `REAL @variable` values were represented with only 2 digits when converted to strings.
- Fixed problem that client "hung" when `LOAD TABLE FROM MASTER` failed.
- `myisamchk --fast --force` will no longer repair tables that only had the open count wrong.
- Added functions to handle symbolic links to make life easier in 4.0.
- We are now using the `-lcma` thread library on HP-UX 10.20 so that MySQL will be more stable on HP-UX.
- Fixed problem with `IF()` and number of decimals in the result.
- Fixed date-part extraction functions to work with dates where day and/or month is 0.
- Extended argument length in option files from 256 to 512 chars.
- Fixed problem with shutdown when `INSERT DELAYED` was waiting for a `LOCK TABLE`.
- Fixed core dump bug in `InnoDB` when tablespace was full.
- Fixed problem with `MERGE` tables and big tables (> 4G) when using `ORDER BY`.

D.4.22 Alterações na distribuição 3.23.38 (09 May 2001)

- Fixed a bug when `SELECT` from `MERGE` table sometimes results in incorrectly ordered rows.
- Fixed a bug in `REPLACE()` when using the `ujis` character set.
- Applied Sleepycat `BDB` patches 3.2.9.1 and 3.2.9.2.
- Added `--skip-stack-trace` option to `mysqld`.
- `CREATE TEMPORARY` now works with `InnoDB` tables.

- InnoDB now promotes sub keys to whole keys.
- Added option CONCURRENT to LOAD DATA.
- Better error message when slave max_allowed_packet is too low to read a very long log event from the master.
- Fixed bug when too many rows where removed when using SELECT DISTINCT ... HAVING.
- SHOW CREATE TABLE now returns TEMPORARY for temporary tables.
- Added Rows_examined to slow query log.
- Fixed problems with function returning empty string when used together with a group function and a WHERE that didn't match any rows.
- New program mysqlcheck.
- Added database name to output for administrative commands like CHECK, REPAIR, OPTIMIZE.
- Lots of portability fixes for InnoDB.
- Changed optimiser so that queries like SELECT * FROM tbl_name,tbl_name2 ... ORDER BY key_part1 LIMIT row_count will use index on key_part1 instead of filesort.
- Fixed bug when doing LOCK TABLE to_table WRITE,...; INSERT INTO to_table... SELECT ... when to_table was empty.
- Fixed bug with LOCK TABLE and BDB tables.

D.4.23 Alterações na distribuição 3.23.37 (17 Apr 2001)

- Fixed a bug when using MATCH() in HAVING clause.
- Fixed a bug when using HEAP tables with LIKE.
- Added --mysql-version option to safe_mysqld
- Changed INNOBASE to InnoDB (because the INNOBASE name was already used). All configure options and mysqld start options now use innodb instead of innobase. This means that before upgrading to this version, you have to change any configuration files where you have used innobase options!
- Fixed bug when using indexes on CHAR(255) NULL columns.
- Slave thread will now be started even if master-host is not set, as long as server-id is set and valid 'master.info' is present.
- Partial updates (terminated with kill) are now logged with a special error code to the binary log. Slave will refuse to execute them if the error code indicates the update was terminated abnormally, and will have to be recovered with SET SQL_SLAVE_SKIP_COUNTER=1; SLAVE START after a manual sanity check/correction of data integrity.
- Fixed bug that erroneously logged a drop of internal temporary table on thread termination to the binary log — this bug affected replication.
- Fixed a bug in REGEXP on 64-bit machines.
- UPDATE and DELETE with WHERE unique_key_part IS NULL didn't update/delete all rows.
- Disabled INSERT DELAYED for tables that support transactions.

- Fixed bug when using date functions on `TEXT`/`BLOB` column with wrong date format.
- UDFs now also work on Windows. (Patch by Ralph Mason.)
- Fixed bug in `ALTER TABLE` and `LOAD DATA INFILE` that disabled key-sorting. These commands should now be faster in most cases.
- Fixed performance bug where reopened tables (tables that had been waiting for `FLUSH` or `REPAIR`) would not use indexes for the next query.
- Fixed problem with `ALTER TABLE` to `InnoDB` tables on FreeBSD.
- Added `mysqld` variables `myisam_max_sort_file_size` and `myisam_max_extra_sort_file_size`.
- Initialise signals early to avoid problem with signals in `InnoDB`.
- Applied patch for the `tis620` character set to make comparisons case-independent and to fix a bug in `LIKE` for this character set. **Note**: All tables that uses the `tis620` character set must be fixed with `myisamchk -r` or `REPAIR TABLE` !
- Added `--skip-safemalloc` option to `mysqld`.

D.4.24 Alterações na distribuição 3.23.36 (27 Mar 2001)

- Fixed a bug that allowed use of database names containing a '`.`' character. This fixes a serious security issue when `mysqld` is run as root.
- Fixed bug when thread creation failed (could happen when doing a **lot** of connections in a short time).
- Fixed some problems with `FLUSH TABLES` and `TEMPORARY` tables. (Problem with freeing the key cache and error `Can't reopen table...`.)
- Fixed a problem in `InnoDB` with other character sets than `latin1` and another problem when using many columns.
- Fixed bug that caused a core dump when using a very complex query involving `DISTINCT` and summary functions.
- Added `SET TRANSACTION ISOLATION LEVEL ...`
- Added `SELECT ... FOR UPDATE`.
- Fixed bug where the number of affected rows was not returned when MySQL was compiled without transaction support.
- Fixed a bug in `UPDATE` where keys weren't always used to find the rows to be updated.
- Fixed a bug in `CONCAT_WS()` where it returned incorrect results.
- Changed `CREATE ... SELECT` and `INSERT ... SELECT` to not allow concurrent inserts as this could make the binary log hard to repeat. (Concurrent inserts are enabled if you are not using the binary or update log.)
- Changed some macros to be able to use fast mutex with `glibc` 2.2.

D.4.25 Alterações na distribuição 3.23.35 (15 Mar 2001)

- Fixed newly introduced bug in `ORDER BY`.
- Fixed wrong define `CLIENT_TRANSACTIONS`.

- Fixed bug in `SHOW VARIABLES` when using `INNOBASE` tables.
- Setting and using user variables in `SELECT DISTINCT` didn't work.
- Tuned `SHOW ANALYZE` for small tables.
- Fixed handling of arguments in the benchmark script `run-all-tests`.

D.4.26 Alterações na distribuição 3.23.34a

- Added extra files to the distribution to allow `INNOBASE` support to be compiled.

D.4.27 Alterações na distribuição 3.23.34 (10 Mar 2001)

- Added the `INNOBASE` storage engine and the `BDB` storage engine to the MySQL source distribution.
- Updated the documentation about `GEMINI` tables.
- Fixed a bug in `INSERT DELAYED` that caused threads to hang when inserting `NULL` into an `AUTO_INCREMENT` column.
- Fixed a bug in `CHECK TABLE` / `REPAIR TABLE` that could cause a thread to hang.
- `REPLACE` will not replace a row that conflicts with an `AUTO_INCREMENT` generated key.
- `mysqld` now only sets `CLIENT_TRANSACTIONS` in `mysql->server_capabilities` if the server supports a transaction-safe storage engine.
- Fixed `LOAD DATA INFILE` to allow numeric values to be read into `ENUM` and `SET` columns.
- Improved error diagnostic for slave thread exit.
- Fixed bug in `ALTER TABLE ... ORDER BY`.
- Added `max_user_connections` variable to `mysqld`.
- Limit query length for replication by `max_allowed_packet`, not the arbitrary limit of 4 MB.
- Allow space around `=` in argument to `--set-variable`.
- Fixed problem in automatic repair that could leave some threads in state `Waiting for table`.
- `SHOW CREATE TABLE` now displays the `UNION=()` for `MERGE` tables.
- `ALTER TABLE` now remembers the old `UNION=()` definition.
- Fixed bug when replicating timestamps.
- Fixed bug in bidirectional replication.
- Fixed bug in the `BDB` storage engine that occurred when using an index on multi-part key where a key part may be `NULL`.
- Fixed `MAX()` optimization on sub-key for `BDB` tables.
- Fixed problem where garbage results were returned when using `BDB` tables and `BLOB` or `TEXT` fields when joining many tables.
- Fixed a problem with `BDB` tables and `TEXT` columns.
- Fixed bug when using a `BLOB` key where a const row wasn't found.

- Fixed that `mysqlbinlog` writes the timestamp value for each query. This ensures that one gets same values for date functions like `NOW()` when using `mysqlbinlog` to pipe the queries to another server.
- Allow `--skip-gemini`, `--skip-bdb`, and `--skip-innodb` options to be specified when invoking `mysqld`, even if these storage engines are not compiled in to `mysqld`.
- One can now do `GROUP BY ... DESC`.
- Fixed a deadlock in the `SET` code, when one ran `SET @foo=bar`, where `bar` is a column reference, an error was not properly generated.

D.4.28 Alterações na distribuição 3.23.33 (09 Feb 2001)

- Fixed DNS lookups not to use the same mutex as the hostname cache. This will enable known hosts to be quickly resolved even if a DNS lookup takes a long time.
- Added `--character-sets-dir` option to `myisampack`.
- Removed warnings when running `REPAIR TABLE ... EXTENDED`.
- Fixed a bug that caused a core dump when using `GROUP BY` on an alias, where the alias was the same as an existing column name.
- Added `SEQUENCE()` as an example UDF function.
- Changed `mysql_install_db` to use `BINARY` for `CHAR` columns in the privilege tables.
- Changed `TRUNCATE tbl_name` to `TRUNCATE TABLE tbl_name` to use the same syntax as Oracle. Until 4.0 we will also allow `TRUNCATE tbl_name` to not crash old code.
- Fixed "no found rows" bug in `MyISAM` tables when a `BLOB` was first part of a multi-part key.
- Fixed bug where `CASE` didn't work with `GROUP BY`.
- Added `--sort-recover` option to `myisamchk`.
- `myisamchk -S` and `OPTIMIZE TABLE` now work on Windows.
- Fixed bug when using `DISTINCT` on results from functions that referred to a group function, like:

```
SELECT a, DISTINCT SEC_TO_TIME(SUM(a))
FROM tbl_name GROUP BY a, b;
```

- Fixed buffer overrun in `libmysqlclient` library. Fixed bug in handling `STOP` event after `ROTATE` event in replication.
- Fixed another buffer overrun in `DROP DATABASE`.
- Added `Table_locks_immediate` and `Table_locks_waited` status variables.
- Fixed bug in replication that broke slave server start with existing '`master.info`'. This fixes a bug introduced in 3.23.32.
- Added `SET SQL_SLAVE_SKIP_COUNTER=n` command to recover from replication glitches without a full database copy.
- Added `max_binlog_size` variable; the binary log will be rotated automatically when the size crosses the limit.
- Added `Last_Error`, `Last_Errno`, and `Slave_skip_counter` variables to `SHOW SLAVE STATUS`.

- Fixed bug in `MASTER_POS_WAIT()` function.
- Execute core dump handler on `SIGILL`, and `SIGBUS` in addition to `SIGSEGV`.
- On x86 Linux, print the current query and thread (connection) id, if available, in the core dump handler.
- Fixed several timing bugs in the test suite.
- Extended `mysqltest` to take care of the timing issues in the test suite.
- `ALTER TABLE` can now be used to change the definition for a `MERGE` table.
- Fixed creation of `MERGE` tables on Windows.
- Portability fixes for OpenBSD and OS/2.
- Added `--temp-pool` option to `mysqld`. Using this option will cause most temporary files created to use a small set of names, rather than a unique name for each new file. This is to work around a problem in the Linux kernel dealing with creating a bunch of new files with different names. With the old behaviour, Linux seems to "leak" memory, as it's being allocated to the directory entry cache instead of the disk cache.

D.4.29 Alterações na distribuição 3.23.32 (22 Jan 2001: Production)

- Changed code to get around compiler bug in Compaq C++ on OSF/1, that broke `BACKUP`, `RESTORE`, `CHECK`, `REPAIR`, and `ANALYZE TABLE`.
- Added option `FULL` to `SHOW COLUMNS`. Now we show the privilege list for the columns only if this option is given.
- Fixed bug in `SHOW LOGS` when there weren't any `BDB` logs.
- Fixed a timing problem in replication that could delay sending an update to the client until a new update was done.
- Don't convert field names when using `mysql_list_fields()`. This is to keep this code compatible with `SHOW FIELDS`.
- `MERGE` tables didn't work on Windows.
- Fixed problem with `SET PASSWORD=...` on Windows.
- Added missing '`my_config.h`' to RPM distribution.
- `TRIM("foo" from "foo")` didn't return an empty string.
- Added `--with-version-suffix` option to `configure`.
- Fixed core dump when client aborted connection without `mysql_close()`.
- Fixed a bug in `RESTORE TABLE` when trying to restore from a non-existent directory.
- Fixed a bug which caused a core dump on the slave when replicating `SET PASSWORD`.
- Added `MASTER_POS_WAIT()`.

D.4.30 Alterações na distribuição 3.23.31 (17 Jan 2001)

- The test suite now tests all reachable `BDB` interface code. During testing we found and fixed many errors in the interface code.
- Using `HAVING` on an empty table could produce one result row when it shouldn't.

- Fixed the MySQL RPM so it no longer depends on Perl5.
- Fixed some problems with `HEAP` tables on Windows.
- `SHOW TABLE STATUS` didn't show correct average row length for tables larger than 4G.
- `CHECK TABLE ... EXTENDED` didn't check row links for fixed size tables.
- Added option `MEDIUM` to `CHECK TABLE`.
- Fixed problem when using `DECIMAL()` keys on negative numbers.
- `HOUR()` (and some other `TIME` functions) on a `CHAR` column always returned `NULL`.
- Fixed security bug in something (please upgrade if you are using an earlier MySQL 3.23 version).
- Fixed buffer overflow bug when writing a certain error message.
- Added usage of `setrlimit()` on Linux to get `-O --open-files-limit=#` to work on Linux.
- Added `bdb_version` variable to `mysqld`.
- Fixed bug when using expression of type:

  ```
  SELECT ... FROM t1 LEFT JOIN t2 ON (t1.a=t2.a) WHERE t1.a=t2.a
  ```

  In this case the test in the `WHERE` clause was wrongly optimised away.
- Fixed bug in `MyISAM` when deleting keys with possible `NULL` values, but the first key-column was not a prefix-compressed text column.
- Fixed `mysql.server` to read the `[mysql.server]` option file group rather than the `[mysql_server]` group.
- Fixed `safe_mysqld` and `mysql.server` to also read the `server` option section.
- Added `Threads_created` status variable to `mysqld`.

D.4.31 Alterações na distribuição 3.23.30 (04 Jan 2001)

- Added `SHOW OPEN TABLES` command.
- Fixed that `myisamdump` works against old `mysqld` servers.
- Fixed `myisamchk -k#` so that it works again.
- Fixed a problem with replication when the binary log file went over 2G on 32-bit systems.
- `LOCK TABLES` will now automatically start a new transaction.
- Changed `BDB` tables to not use internal subtransactions and reuse open files to get more speed.
- Added `--mysqld=#` option to `safe_mysqld`.
- Allow hex constants in the `--fields-*-by` and `--lines-terminated-by` options to `mysqldump` and `mysqlimport`. By Paul DuBois.
- Added `--safe-show-database` option to `mysqld`.
- Added `have_bdb`, `have_gemini`, `have_innobase`, `have_raid` and `have_openssl` to `SHOW VARIABLES` to make it easy to test for supported extensions.
- Added `--open-files-limit` option to `mysqld`.
- Changed `--open-files` option to `--open-files-limit` in `safe_mysqld`.

- Fixed a bug where some rows were not found with `HEAP` tables that had many keys.
- Fixed that `--bdb-no-sync` works.
- Changed `--bdb-recover` to `--bdb-no-recover` as recover should be on by default.
- Changed the default number of `BDB` locks to 10000.
- Fixed a bug from 3.23.29 when allocating the shared structure needed for `BDB` tables.
- Changed `mysqld_multi.sh` to use configure variables. Patch by Christopher McCrory.
- Added fixing of include files for Solaris 2.8.
- Fixed bug with `--skip-networking` on Debian Linux.
- Fixed problem that some temporary files where reported as having the name `UNOPENED` in error messages.
- Fixed bug when running two simultaneous `SHOW LOGS` queries.

D.4.32 Alterações na distribuição 3.23.29 (16 Dec 2000)

- Configure updates for Tru64, large file support, and better TCP wrapper support. By Albert Chin-A-Young.
- Fixed bug in `<=>` operator.
- Fixed bug in `REPLACE` with `BDB` tables.
- `LPAD()` and `RPAD()` will shorten the result string if it's longer than the length argument.
- Added `SHOW LOGS` command.
- Remove unused `BDB` logs on shutdown.
- When creating a table, put `PRIMARY` keys first, followed by `UNIQUE` keys.
- Fixed a bug in `UPDATE` involving multi-part keys where one specified all key parts both in the update and the `WHERE` part. In this case MySQL could try to update a record that didn't match the whole `WHERE` part.
- Changed drop table to first drop the tables and then the '`.frm`' file.
- Fixed a bug in the hostname cache which caused `mysqld` to report the hostname as `''` in some error messages.
- Fixed a bug with `HEAP` type tables; the variable `max_heap_table_size` wasn't used. Now either `MAX_ROWS` or `max_heap_table_size` can be used to limit the size of a `HEAP` type table.
- Changed the default server-id to 1 for masters and 2 for slaves to make it easier to use the binary log.
- Renamed `bdb_lock_max` variable to `bdb_max_lock`.
- Added support for `AUTO_INCREMENT` on sub-fields for `BDB` tables.
- Added `ANALYZE` of `BDB` tables.
- In `BDB` tables, we now store the number of rows; this helps to optimise queries when we need an approximation of the number of rows.
- If we get an error in a multi-row statement, we now only roll back the last statement, not the entire transaction.
- If you do a `ROLLBACK` when you have updated a non-transactional table you will get an error as a warning.

- Added `--bdb-shared-data` option to `mysqld`.
- Added `Slave_open_temp_tables` status variable to `mysqld`
- Added `binlog_cache_size` and `max_binlog_cache_size` variables to `mysqld`.
- `DROP TABLE`, `RENAME TABLE`, `CREATE INDEX` and `DROP INDEX` are now transaction endpoints.
- If you do a `DROP DATABASE` on a symbolically linked database, both the link and the original database is deleted.
- Fixed `DROP DATABASE` to work on OS/2.
- Fixed bug when doing a `SELECT DISTINCT ... table1 LEFT JOIN table2 ...` when `table2` was empty.
- Added `--abort-slave-event-count` and `--disconnect-slave-event-count` options to `mysqld` for debugging and testing of replication.
- Fixed replication of temporary tables. Handles everything except slave server restart.
- `SHOW KEYS` now shows whether key is `FULLTEXT`.
- New script `mysqld_multi`. Veja Seção 4.8.3 [mysqld_multi], Página 333.
- Added new script, `mysql-multi.server.sh`. Thanks to Tim Bunce `Tim.Bunce@ig.co.uk` for modifying `mysql.server` to easily handle hosts running many `mysqld` processes.
- `safe_mysqld`, `mysql.server`, and `mysql_install_db` have been modified to use `mysql_print_defaults` instead of various hacks to read the '`my.cnf`' files. In addition, the handling of various paths has been made more consistent with how `mysqld` handles them by default.
- Automatically remove Berkeley DB transaction logs that no longer are in use.
- Fixed bug with several `FULLTEXT` indexes in one table.
- Added a warning if number of rows changes on `REPAIR/OPTIMIZE`.
- Applied patches for OS/2 by `Yuri Dario`.
- `FLUSH TABLES tbl_name` didn't always flush the index tree to disk properly.
- `--bootstrap` is now run in a separate thread. This fixes a problem that caused `mysql_install_db` to core dump on some Linux machines.
- Changed `mi_create()` to use less stack space.
- Fixed bug with optimiser trying to over-optimise `MATCH()` when used with `UNIQUE` key.
- Changed `crash-me` and the MySQL benchmarks to also work with FrontBase.
- Allow `RESTRICT` and `CASCADE` after `DROP TABLE` to make porting easier.
- Reset status variable which could cause problem if one used `--slow-log`.
- Added `connect_timeout` variable to `mysql` and `mysqladmin`.
- Added `connect-timeout` as an alias for `timeout` for option files read by `mysql_options()`.

D.4.33 Alterações na distribuição 3.23.28 (22 Nov 2000: Gamma)

- Added new options `--pager[=...]`, `--no-pager`, `--tee=...` and `--no-tee` to the `mysql` client. The new corresponding interactive commands are `pager`, `nopager`, `tee`

and `notee`. Veja Seção 4.9.2 [`mysql`], Página 347, `mysql --help` and the interactive help for more information.

- Fixed crash when automatic repair of `MyISAM` table failed.
- Fixed a major performance bug in the table locking code when one constantly had a lot of `SELECT`, `UPDATE` and `INSERT` statements running. The symptom was that the `UPDATE` and `INSERT` queries were locked for a long time while new `SELECT` statements were executed before the updates.
- When reading `options_files` with `mysql_options()` the `return-found-rows` option was ignored.
- One can now specify `interactive-timeout` in the option file that is read by `mysql_options()`. This makes it possible to force programs that run for a long time (like `mysqlhotcopy`) to use the `interactive_timeout` time instead of the `wait_timeout` time.
- Added to the slow query log the time and the user name for each logged query. If you are using `--log-long-format` then also queries that do not use an index are logged, even if the query takes less than `long_query_time` seconds.
- Fixed a problem in `LEFT JOIN` which caused all columns in a reference table to be `NULL`.
- Fixed a problem when using `NATURAL JOIN` without keys.
- Fixed a bug when using a multi-part keys where the first part was of type `TEXT` or `BLOB`.
- `DROP` of temporary tables wasn't stored in the update/binary log.
- Fixed a bug where `SELECT DISTINCT * ... LIMIT row_count` only returned one row.
- Fixed a bug in the assembler code in `strstr()` for SPARC and cleaned up the '`global.h`' header file to avoid a problem with bad aliasing with the compiler submitted with Red Hat 7.0. (Reported by Trond Eivind Glomsrød)
- The `--skip-networking` option now works properly on NT.
- Fixed a long outstanding bug in the `ISAM` tables when a row with a length of more than 65K was shortened by a single byte.
- Fixed a bug in `MyISAM` when running multiple updating processes on the same table.
- Allow one to use `FLUSH TABLE tbl_name`.
- Added `--replicate-ignore-table`, `--replicate-do-table`, `--replicate-wild-ignore-table`, and `--replicate-wild-do-table` options to `mysqld`.
- Changed all log files to use our own `IO_CACHE` mechanism instead of `FILE` to avoid OS problems when there are many files open.
- Added `--open-files` and `--timezone` options to `safe_mysqld`.
- Fixed a fatal bug in `CREATE TEMPORARY TABLE ... SELECT ...`.
- Fixed a problem with `CREATE TABLE ... SELECT NULL`.
- Added variables `large_file_support`,`net_read_timeout`, `net_write_timeout` and `query_buffer_size` to `SHOW VARIABLES`.
- Added status variables `created_tmp_files` and `sort_merge_passes` to `SHOW STATUS`.
- Fixed a bug where we didn't allow an index name after the `FOREIGN KEY` definition.
- Added `TRUNCATE table_name` as a synonym for `DELETE FROM table_name`.

- Fixed a bug in a BDB key compare function when comparing part keys.
- Added bdb_lock_max variable to mysqld.
- Added more tests to the benchmark suite.
- Fixed an overflow bug in the client code when using overly long database names.
- mysql_connect() now aborts on Linux if the server doesn't answer in timeout seconds.
- SLAVE START did not work if you started with --skip-slave-start and had not explicitly run CHANGE MASTER TO.
- Fixed the output of SHOW MASTER STATUS to be consistent with SHOW SLAVE STATUS. (It now has no directory in the log name.)
- Added PURGE MASTER LOGS TO.
- Added SHOW MASTER LOGS.
- Added --safemalloc-mem-limit option to mysqld to simulate memory shortage when compiled with the --with-debug=full option.
- Fixed several core dumps in out-of-memory conditions.
- SHOW SLAVE STATUS was using an uninitialised mutex if the slave had not been started yet.
- Fixed bug in ELT() and MAKE_SET() when the query used a temporary table.
- CHANGE MASTER TO without specifying MASTER_LOG_POS would set it to 0 instead of 4 and hit the magic number in the master binlog.
- ALTER TABLE ... ORDER BY ... syntax added. This will create the new table with the rows in a specific order.

D.4.34 Alterações na distribuição 3.23.27 (24 Oct 2000)

- Fixed a bug where the automatic repair of MyISAM tables sometimes failed when the datafile was corrupt.
- Fixed a bug in SHOW CREATE when using AUTO_INCREMENT columns.
- Changed BDB tables to use new compare function in Berkeley DB 3.2.3.
- You can now use Unix sockets with MIT-pthreads.
- Added the latin5 (turkish) character set.
- Small portability fixes.

D.4.35 Alterações na distribuição 3.23.26 (18 Oct 2000)

- Renamed FLUSH MASTER and FLUSH SLAVE to RESET MASTER and RESET SLAVE.
- Fixed <> to work properly with NULL.
- Fixed a problem with SUBSTRING_INDEX() and REPLACE(). (Patch by Alexander Igonitchev)
- Fix CREATE TEMPORARY TABLE IF NOT EXISTS not to produce an error if the table exists.
- If you don't create a PRIMARY KEY in a BDB table, a hidden PRIMARY KEY will be created.
- Added read-only-key optimization to BDB tables.

- `LEFT JOIN` in some cases preferred a full table scan when there was no `WHERE` clause.
- When using `--log-slow-queries`, don't count the time waiting for a lock.
- Fixed bug in lock code on Windows which could cause the key cache to report that the key file was crashed even if it was okay.
- Automatic repair of `MyISAM` tables if you start `mysqld` with `--myisam-recover`.
- Removed the `TYPE=` keyword from `CHECK` and `REPAIR`. Allow `CHECK` options to be combined. (You can still use `TYPE=`, but this usage is deprecated.)
- Fixed mutex bug in the binary replication log — long update queries could be read only in part by the slave if it did it at the wrong time, which was not fatal, but resulted in a performance-degrading reconnect and a scary message in the error log.
- Changed the format of the binary log — added magic number, server version, binlog version. Added server ID and query error code for each query event.
- Replication thread from the slave now will kill all the stale threads from the same server.
- Long replication user names were not being handled properly.
- Added `--replicate-rewrite-db` option to `mysqld`.
- Added `--skip-slave-start` option to `mysqld`.
- Updates that generated an error code (such as `INSERT INTO foo(some_key) values (1),(1)`) erroneously terminated the slave thread.
- Added optimization of queries where `DISTINCT` is only used on columns from some of the tables.
- Allow floating-point numbers where there is no sign after the exponent (like `1e1`).
- `SHOW GRANTS` didn't always show all column grants.
- Added `--default-extra-file=#` option to all MySQL clients.
- Columns referenced in `INSERT` statements now are initialised properly.
- `UPDATE` didn't always work when used with a range on a timestamp that was part of the key that was used to find rows.
- Fixed a bug in `FULLTEXT` index when inserting a `NULL` column.
- Changed to use `mkstemp()` instead of `tempnam()`. Based on a patch from John Jones.

D.4.36 Alterações na distribuição 3.23.25 (29 Sep 2000)

- Fixed that `databasename` works as second argument to `mysqlhotcopy`.
- The values for the `UMASK` and `UMASK_DIR` environment variables now can be specified in octal by beginning the value with a zero.
- Added `RIGHT JOIN`. This makes `RIGHT` a reserved word.
- Added `@@IDENTITY` as a synonym for `LAST_INSERT_ID()`. (This is for MSSQL compatibility.)
- Fixed a bug in `myisamchk` and `REPAIR` when using `FULLTEXT` index.
- `LOAD DATA INFILE` now works with FIFOs. (Patch by Toni L. Harbaugh-Blackford.)
- `FLUSH LOGS` broke replication if you specified a log name with an explicit extension as the value of the `log-bin` option.

- Fixed a bug in `MyISAM` with packed multi-part keys.
- Fixed crash when using `CHECK TABLE` on Windows.
- Fixed a bug where `FULLTEXT` index always used the `koi8_ukr` character set.
- Fixed privilege checking for `CHECK TABLE`.
- The `MyISAM` repair/reindex code didn't use the `--tmpdir` option for its temporary files.
- Added `BACKUP TABLE` and `RESTORE TABLE`.
- Fixed core dump on `CHANGE MASTER TO` when the slave did not have the master to start with.
- Fixed incorrect `Time` in the processlist for `Connect` of the slave thread.
- The slave now logs when it connects to the master.
- Fixed a core dump bug when doing `FLUSH MASTER` if you didn't specify a filename argument to `--log-bin`.
- Added missing '`ha_berkeley.x`' files to the MySQL Windows distribution.
- Fixed some mutex bugs in the log code that could cause thread blocks if new log files couldn't be created.
- Added lock time and number of selected processed rows to slow query log.
- Added `--memlock` option to `mysqld` to lock `mysqld` in memory on systems with the `mlockall()` call (as in Solaris).
- `HEAP` tables didn't use keys properly. (Bug from 3.23.23.)
- Added better support for `MERGE` tables (keys, mapping, creation, documentation...). Veja Seção 7.2 [MERGE], Página 638.
- Fixed bug in `mysqldump` from 3.23 which caused some `CHAR` columns not to be quoted.
- Merged `analyze`, `check`, `optimize` and repair code.
- `OPTIMIZE TABLE` is now mapped to `REPAIR` with statistics and sorting of the index tree. This means that for the moment it only works on `MyISAM` tables.
- Added a pre-alloced block to root_malloc to get fewer mallocs.
- Added a lot of new statistics variables.
- Fixed `ORDER BY` bug with `BDB` tables.
- Removed warning that `mysqld` couldn't remove the '`.pid`' file under Windows.
- Changed `--log-isam` to log `MyISAM` tables instead of isam tables.
- Fixed `CHECK TABLE` to work on Windows.
- Added file mutexes to make `pwrite()` safe on Windows.

D.4.37 Alterações na distribuição 3.23.24 (08 Sep 2000)

- Added `created_tmp_disk_tables` variable to `mysqld`.
- To make it possible to reliably dump and restore tables with `TIMESTAMP(X)` columns, MySQL now reports columns with `X` other than 14 or 8 to be strings.
- Changed sort order for `latin1` as it was before MySQL Version 3.23.23. Any table that was created or modified with 3.23.22 must be repaired if it has `CHAR` columns that may contain characters with ASCII values greater than 128!

- Fixed small memory leak introduced from 3.23.22 when creating a temporary table.
- Fixed problem with `BDB` tables and reading on a unique (not primary) key.
- Restored the `win1251` character set (it's now only marked deprecated).

D.4.38 Alterações na distribuição 3.23.23 (01 Sep 2000)

- Changed sort order for 'German'; all tables created with 'German' sortorder must be repaired with `REPAIR TABLE` or `myisamchk` before use!
- Added `--core-file` option to `mysqld` to get a core file on Linux if `mysqld` dies on the `SIGSEGV` signal.
- MySQL client `mysql` now starts with option `--no-named-commands` (`-g`) by default. This option can be disabled with `--enable-named-commands` (`-G`). This may cause incompatibility problems in some cases, for example, in SQL scripts that use named commands without a semicolon, etc.! Long format commands still work from the first line.
- Fixed a problem when using many pending `DROP TABLE` statements at the same time.
- Optimizer didn't use keys properly when using `LEFT JOIN` on an empty table.
- Added shorter help text when invoking `mysqld` with incorrect options.
- Fixed non-fatal `free()` bug in `mysqlimport`.
- Fixed bug in `MyISAM` index handling of `DECIMAL/NUMERIC` keys.
- Fixed a bug in concurrent insert in `MyISAM` tables. In some contexts, usage of `MIN(key_part)` or `MAX(key_part)` returned an empty set.
- Updated `mysqlhotcopy` to use the new `FLUSH TABLES table_list` syntax. Only tables which are being backed up are flushed now.
- Changed behaviour of `--enable-thread-safe-client` so that both non-threaded (`-lmysqlclient`) and threaded (`-lmysqlclient_r`) libraries are built. Users who linked against a threaded `-lmysqlclient` will need to link against `-lmysqlclient_r` now.
- Added atomic `RENAME TABLE` command.
- Don't count `NULL` values in `COUNT(DISTINCT ...)`.
- Changed `ALTER TABLE`, `LOAD DATA INFILE` on empty tables and `INSERT ... SELECT ...` on empty tables to create non-unique indexes in a separate batch with sorting. This will make the above calls much faster when you have many indexes.
- `ALTER TABLE` now logs the first used insert_id correctly.
- Fixed crash when adding a default value to a `BLOB` column.
- Fixed a bug with `DATE_ADD/DATE_SUB` where it returned a datetime instead of a date.
- Fixed a problem with the thread cache which made some threads show up as `***DEAD***` in `SHOW PROCESSLIST`.
- Fixed a lock in our thr_rwlock code, which could make selects that run at the same time as concurrent inserts crash. This only affects systems that don't have the `pthread_rwlock_rdlock` code.
- When deleting rows with a non-unique key in a `HEAP` table, all rows weren't always deleted.

- Fixed bug in range optimiser for `HEAP` tables for searches on a part index.
- Fixed `SELECT` on part keys to work with `BDB` tables.
- Fixed `INSERT INTO bdb_table ... SELECT` to work with `BDB` tables.
- `CHECK TABLE` now updates key statistics for the table.
- `ANALYZE TABLE` will now only update tables that have been changed since the last `ANALYZE`. Note that this is a new feature and tables will not be marked to be analysed until they are updated in any way with 3.23.23 or newer. For older tables, you have to do `CHECK TABLE` to update the key distribution.
- Fixed some minor privilege problems with `CHECK`, `ANALYZE`, `REPAIR` and `SHOW CREATE` commands.
- Added `CHANGE MASTER TO` statement.
- Added `FAST`, `QUICK EXTENDED` check types to `CHECK TABLES`.
- Changed `myisamchk` so that `--fast` and `--check-only-changed` are also honored with `--sort-index` and `--analyze`.
- Fixed fatal bug in `LOAD TABLE FROM MASTER` that did not lock the table during index re-build.
- `LOAD DATA INFILE` broke replication if the database was excluded from replication.
- More variables in `SHOW SLAVE STATUS` and `SHOW MASTER STATUS`.
- `SLAVE STOP` now will not return until the slave thread actually exits.
- Full-text search via the `MATCH()` function and `FULLTEXT` index type (for `MyISAM` files). This makes `FULLTEXT` a reserved word.

D.4.39 Alterações na distribuição 3.23.22 (31 Jul 2000)

- Fixed that `lex_hash.h` is created properly for each MySQL distribution.
- Fixed that `MASTER` and `COLLECTION` are not reserved words.
- The log generated by `--slow-query-log` didn't contain the whole queries.
- Fixed that open transactions in `BDB` tables are rolled back if the connection is closed unexpectedly.
- Added workaround for a bug in `gcc` 2.96 (intel) and `gcc` 2.9 (IA-64) in `gen_lex_hash.c`.
- Fixed memory leak in the client library when using `host=` in the '`my.cnf`' file.
- Optimized functions that manipulate the hours/minutes/seconds.
- Fixed bug when comparing the result of `DATE_ADD()`/`DATE_SUB()` against a number.
- Changed the meaning of `-F, --fast` for `myisamchk`. Added `-C, --check-only-changed` option to `myisamchk`.
- Added `ANALYZE tbl_name` to update key statistics for tables.
- Changed binary items `0x...` to be regarded as integers by default.
- Fix for SCO and `SHOW PROCESSLIST`.
- Added `auto-rehash` on reconnect for the `mysql` client.
- Fixed a newly introduced bug in `MyISAM`, where the index file couldn't get bigger than 64M.
- Added `SHOW MASTER STATUS` and `SHOW SLAVE STATUS`.

D.4.40 Alterações na distribuição 3.23.21

- Added `mysql_character_set_name()` function to the MySQL C API.
- Made the update log ASCII 0 safe.
- Added the `mysql_config` script.
- Fixed problem when using `<` or `>` with a char column that was only partly indexed.
- One would get a core dump if the log file was not readable by the MySQL user.
- Changed `mysqladmin` to use `CREATE DATABASE` and `DROP DATABASE` statements instead of the old deprecated API calls.
- Fixed `chown` warning in `safe_mysqld`.
- Fixed a bug in `ORDER BY` that was introduced in 3.23.19.
- Only optimise the `DELETE FROM tbl_name` to do a drop+create of the table if we are in `AUTOCOMMIT` mode (needed for `BDB` tables).
- Added extra checks to avoid index corruption when the `ISAM`/`MyISAM` index files get full during an `INSERT`/`UPDATE`.
- `myisamchk` didn't correctly update row checksum when used with `-ro` (this only gave a warning in subsequent runs).
- Fixed bug in `REPAIR TABLE` so that it works with tables without indexes.
- Fixed buffer overrun in `DROP DATABASE`.
- `LOAD TABLE FROM MASTER` is sufficiently bug-free to announce it as a feature.
- `MATCH` and `AGAINST` are now reserved words.

D.4.41 Alterações na distribuição 3.23.20

- Fixed bug in 3.23.19; `DELETE FROM tbl_name` removed the '`.frm`' file.
- Added `SHOW CREATE TABLE`.

D.4.42 Alterações na distribuição 3.23.19

- Changed copyright for all files to `GPL` for the server code and utilities and to LGPL for the client libraries. See `http://www.fsf.org/licenses/`.
- Fixed bug where all rows matching weren't updated on a `MyISAM` table when doing update based on key on a table with many keys and some key changed values.
- The Linux MySQL RPMs and binaries are now statically linked with a linuxthread version that has faster mutex handling when used with MySQL.
- `ORDER BY` can now use `REF` keys to find subsets of the rows that need to be sorted.
- Changed name of `print_defaults` program to `my_print_defaults` to avoid name confusion.
- Fixed `NULLIF()` to work as required by SQL-99.
- Added `net_read_timeout` and `net_write_timeout` as startup parameters to `mysqld`.
- Fixed bug that destroyed index when doing `myisamchk --sort-records` on a table with prefix compressed index.

- Added `pack_isam` and `myisampack` to the standard MySQL distribution.
- Added the syntax `BEGIN WORK` (the same as `BEGIN`).
- Fixed core dump bug when using `ORDER BY` on a `CONV()` expression.
- Added `LOAD TABLE FROM MASTER`.
- Added `FLUSH MASTER` and `FLUSH SLAVE`.
- Fixed big/little endian problem in the replication.

D.4.43 Alterações na distribuição 3.23.18

- Fixed a problem from 3.23.17 when choosing character set on the client side.
- Added `FLUSH TABLES WITH READ LOCK` to make a global lock suitable for making a copy of MySQL datafiles.
- `CREATE TABLE ... SELECT ... PROCEDURE` now works.
- Internal temporary tables will now use compressed index when using `GROUP BY` on `VARCHAR/CHAR` columns.
- Fixed a problem when locking the same table with both a `READ` and a `WRITE` lock.
- Fixed problem with `myisamchk` and `RAID` tables.

D.4.44 Alterações na distribuição 3.23.17

- Fixed a bug in `FIND_IN_SET()` when the first argument was `NULL`.
- Added table locks to Berkeley DB.
- Fixed a bug with `LEFT JOIN` and `ORDER BY` where the first table had only one matching row.
- Added 4 sample '`my.cnf`' example files in the '`support-files`' directory.
- Fixed `duplicated key` problem when doing big `GROUP BY` operations. (This bug was probably introduced in 3.23.15.)
- Changed syntax for `INNER JOIN` to match SQL-99.
- Added `NATURAL JOIN` syntax.
- A lot of fixes in the `BDB` interface.
- Added handling of `--no-defaults` and `--defaults-file` to `safe_mysqld.sh` and `mysql_install_db.sh`.
- Fixed bug in reading compressed tables with many threads.
- Fixed that `USE INDEX` works with `PRIMARY` keys.
- Added `BEGIN` statement to start a transaction in `AUTOCOMMIT` mode.
- Added support for symbolic links for Windows.
- Changed protocol to let client know if the server is in `AUTOCOMMIT` mode and if there is a pending transaction. If there is a pending transaction, the client library will give an error before reconnecting to the server to let the client know that the server did a rollback. The protocol is still backward-compatible with old clients.
- `KILL` now works on a thread that is locked on a 'write' to a dead client.

- Fixed memory leak in the replication slave thread.
- Added new `log-slave-updates` option to `mysqld`, to allow daisy-chaining the slaves.
- Fixed compile error on FreeBSD and other systems where `pthread_t` is not the same as `int`.
- Fixed master shutdown aborting the slave thread.
- Fixed a race condition in `INSERT DELAYED` code when doing `ALTER TABLE`.
- Added deadlock detection sanity checks to `INSERT DELAYED`.

D.4.45 Alterações na distribuição 3.23.16

- Added `SLAVE START` and `SLAVE STOP` statements.
- Added `TYPE=QUICK` option to `CHECK` and to `REPAIR`.
- Fixed bug in `REPAIR TABLE` when the table was in use by other threads.
- Added a thread cache to make it possible to debug MySQL with `gdb` when one does a lot of reconnects. This will also improve systems where you can't use persistent connections.
- Lots of fixes in the Berkeley DB interface.
- `UPDATE IGNORE` will not abort if an update results in a `DUPLICATE_KEY` error.
- Put `CREATE TEMPORARY TABLE` commands in the update log.
- Fixed bug in handling of masked IP numbers in the privilege tables.
- Fixed bug with `delay_key_write` tables and `CHECK TABLE`.
- Added `replicate-do-db` and `replicate-ignore-db` options to `mysqld`, to restrict which databases get replicated.
- Added `SQL_LOG_BIN` option.

D.4.46 Alterações na distribuição 3.23.15 (May 2000: Beta)

- To start `mysqld` as `root`, you must now use the `--user=root` option.
- Added interface to Berkeley DB. (This is not yet functional; play with it at your own risk!)
- Replication between master and slaves.
- Fixed bug that other threads could steal a lock when a thread had a lock on a table and did a `FLUSH TABLES` command.
- Added the `slow_launch_time` variable and the `Slow_launch_threads` status variable to `mysqld`. These can be examined with `mysqladmin variables` and `mysqladmin extended-status`.
- Added functions `INET_NTOA()` and `INET_ATON()`.
- The default type of `IF()` now depends on the second and third arguments and not only on the second argument.
- Fixed case when `myisamchk` could go into a loop when trying to repair a crashed table.
- Don't write `INSERT DELAYED` to update log if `SQL_LOG_UPDATE=0`.
- Fixed problem with `REPLACE` on `HEAP` tables.

- Added possible character sets and time zone to `SHOW VARIABLES` output.
- Fixed bug in locking code that could result in locking problems with concurrent inserts under high load.
- Fixed a problem with `DELETE` of many rows on a table with compressed keys where MySQL scanned the index to find the rows.
- Fixed problem with `CHECK` on table with deleted keyblocks.
- Fixed a bug in reconnect (at the client side) where it didn't free memory properly in some contexts.
- Fixed problems in update log when using `LAST_INSERT_ID()` to update a table with an `AUTO_INCREMENT` key.
- Added `NULLIF()` function.
- Fixed bug when using `LOAD DATA INFILE` on a table with `BLOB/TEXT` columns.
- Optimized `MyISAM` to be faster when inserting keys in sorted order.
- `EXPLAIN SELECT ...` now also prints out whether MySQL needs to create a temporary table or use file sorting when resolving the `SELECT`.
- Added optimization to skip `ORDER BY` parts where the part is a constant expression in the `WHERE` part. Indexes can now be used even if the `ORDER BY` doesn't match the index exactly, as long as all the unused index parts and all the extra `ORDER BY` columns are constants in the `WHERE` clause. Veja Seção 5.4.3 [MySQL indexes], Página 447.
- `UPDATE` and `DELETE` on a whole unique key in the `WHERE` part are now faster than before.
- Changed `RAID_CHUNKSIZE` to be in 1024-byte increments.
- Fixed core dump in `LOAD_FILE(NULL)`.

D.4.47 Alterações na distribuição 3.23.14

- Added `mysql_real_escape_string()` function to the MySQL C API.
- Fixed a bug in `CONCAT()` where one of the arguments was a function that returned a modified argument.
- Fixed a critical bug in `myisamchk`, where it updated the header in the index file when one only checked the table. This confused the `mysqld` daemon if it updated the same table at the same time. Now the status in the index file is only updated if one uses `--update-state`. With older `myisamchk` versions you should use `--read-only` when only checking tables, if there is the slightest chance that the `mysqld` server is working on the table at the same time!
- Fixed that `DROP TABLE` is logged in the update log.
- Fixed problem when searching on `DECIMAL()` key field where the column data contained leading zeros.
- Fix bug in `myisamchk` when the `AUTO_INCREMENT` column isn't the first key.
- Allow `DATETIME` in ISO8601 format: 2000-03-12T12:00:00
- Dynamic character sets. A `mysqld` binary can now handle many different character sets (you can choose which when starting `mysqld`).
- Added command `REPAIR TABLE`.

- Added `mysql_thread_safe()` function to the MySQL C API.
- Added the `UMASK_DIR` environment variable.
- Added `CONNECTION_ID()` function to return the client connection thread ID.
- When using `=` on `BLOB` or `VARCHAR BINARY` keys, where only a part of the column was indexed, the whole column of the result row wasn't compared.
- Fix for `sjis` character set and `ORDER BY`.
- When running in ANSI mode, don't allow columns to be used that aren't in the `GROUP BY` part.

D.4.48 Alterações na distribuição 3.23.13

- Fixed problem when doing locks on the same table more than 2 times in the same `LOCK TABLE` command; this fixed the problem one got when running the test-ATIS test with `--fast` or `--check-only-changed`.
- Added `SQL_BUFFER_RESULT` option to `SELECT`.
- Removed end space from double/float numbers in results from temporary tables.
- Added `CHECK TABLE` command.
- Added changes for `MyISAM` in 3.23.12 that didn't get into the source distribution because of CVS problems.
- Fixed bug so that `mysqladmin shutdown` will wait for the local server to close down.
- Fixed a possible endless loop when calculating timestamp.
- Added `print_defaults` program to the '`.rpm`' files. Removed `mysqlbug` from the client '`.rpm`' file.

D.4.49 Alterações na distribuição 3.23.12 (07 Mar 2000)

- Fixed bug in `MyISAM` involving `REPLACE ... SELECT ...` which could give a corrupted table.
- Fixed bug in `myisamchk` where it incorrectly reset the `AUTO_INCREMENT` value.
- LOTS of patches for Linux Alpha. MySQL now appears to be relatively stable on Alpha.
- Changed `DISTINCT` on `HEAP` temporary tables to use hashed keys to quickly find duplicated rows. This mostly concerns queries of type `SELECT DISTINCT ... GROUP BY ...`. This fixes a problem where not all duplicates were removed in queries of the above type. In addition, the new code is MUCH faster.
- Added patches to make MySQL compile on Mac OS X.
- Added `IF NOT EXISTS` clause to `CREATE DATABASE`.
- Added `--all-databases` and `--databases` options to `mysqldump` to allow dumping of many databases at the same time.
- Fixed bug in compressed `DECIMAL()` index in `MyISAM` tables.
- Fixed bug when storing 0 into a timestamp.
- When doing `mysqladmin shutdown` on a local connection, `mysqladmin` now waits until the PID file is gone before terminating.

- Fixed core dump with some `COUNT(DISTINCT ...)` queries.
- Fixed that `myisamchk` works properly with RAID tables.
- Fixed problem with `LEFT JOIN` and `key_field IS NULL`.
- Fixed bug in `net_clear()` which could give the error `Aborted connection` in the MySQL clients.
- Added options `USE INDEX (key_list)` and `IGNORE INDEX (key_list)` as parameters in `SELECT`.
- `DELETE` and `RENAME` should now work on `RAID` tables.

D.4.50 Alterações na distribuição 3.23.11

- Allow the `ALTER TABLE tbl_name ADD (field_list)` syntax.
- Fixed problem with optimiser that could sometimes use incorrect keys.
- Fixed that `GRANT/REVOKE ALL PRIVILEGES` doesn't affect `GRANT OPTION`.
- Removed extra '`)`' from the output of `SHOW GRANTS`.
- Fixed problem when storing numbers in timestamps.
- Fix problem with timezones that have half hour offsets.
- Allow the syntax `UNIQUE INDEX` in `CREATE` statements.
- `mysqlhotcopy` - fast online hot-backup utility for local MySQL databases. By Tim Bunce.
- New more secure `mysqlaccess`. Thanks to Steve Harvey for this.
- Added `--i-am-a-dummy` and `--safe-updates` options to `mysql`.
- Added `select_limit` and `max_join_size` variables to `mysql`.
- Added `SQL_MAX_JOIN_SIZE` and `SQL_SAFE_UPDATES` options.
- Added `READ LOCAL` lock that doesn't lock the table for concurrent inserts. (This is used by `mysqldump`.)
- Changed that `LOCK TABLES ... READ` doesn't anymore allow concurrent inserts.
- Added `--skip-delay-key-write` option to `mysqld`.
- Fixed security problem in the protocol regarding password checking.
- `_rowid` can now be used as an alias for an integer type unique indexed column.
- Added back blocking of `SIGPIPE` when compiling with `--thread-safe-clients` to make things safe for old clients.

D.4.51 Alterações na distribuição 3.23.10

- Fixed bug in 3.23.9 where memory wasn't properly freed when using `LOCK TABLES`.

D.4.52 Alterações na distribuição 3.23.9

- Fixed problem that affected queries that did arithmetic on group functions.
- Fixed problem with timestamps and `INSERT DELAYED`.
- Fixed that `date_col BETWEEN const_date AND const_date` works.

- Fixed problem when only changing a 0 to `NULL` in a table with `BLOB/TEXT` columns.
- Fixed bug in range optimiser when using many key parts and or on the middle key parts: `WHERE K1=1 and K3=2 and (K2=2 and K4=4 or K2=3 and K4=5)`
- Added `source` command to `mysql` to allow reading of batch files inside the `mysql` client. Original patch by Matthew Vanecek.
- Fixed critical problem with the `WITH GRANT OPTION` option.
- Don't give an unnecessary `GRANT` error when using tables from many databases in the same query.
- Added VIO wrapper (needed for SSL support; by Andrei Errapart and Tõnu Samuel).
- Fixed optimiser problem on `SELECT` when using many overlapping indexes. MySQL should now be able to choose keys even better when there are many keys to choose from.
- Changed optimiser to prefer a range key instead of a ref key when the range key can uses more columns than the ref key (which only can use columns with `=`). For example, the following type of queries should now be faster: `SELECT * from key_part_1=const and key_part_2 > const2`
- Fixed bug that a change of all `VARCHAR` columns to `CHAR` columns didn't change row type from dynamic to fixed.
- Disabled floating-point exceptions for FreeBSD to fix core dump when doing `SELECT FLOOR(POW(2,63))`.
- Renamed `mysqld` startup option from `--delay-key-write` to `--delay-key-write-for-all-tables`.
- Added `read-next-on-key` to `HEAP` tables. This should fix all problems with `HEAP` tables when using non-`UNIQUE` keys.
- Added option to print default arguments to all clients.
- Added `--log-slow-queries` option to `mysqld` to log all queries that take a long time to a separate log file with a time indicating how long the query took.
- Fixed core dump when doing `WHERE key_col=RAND(...)`.
- Fixed optimization bug in `SELECT ... LEFT JOIN ... key_col IS NULL`, when `key_col` could contain `NULL` values.
- Fixed problem with 8-bit characters as separators in `LOAD DATA INFILE`.

D.4.53 Alterações na distribuição 3.23.8 (02 Jan 2000)

- Fixed problem when handling indexfiles larger than 8G.
- Added latest patches to MIT-pthreads for NetBSD.
- Fixed problem with timezones that are < GMT - 11.
- Fixed a bug when deleting packed keys in `NISAM`.
- Fixed problem with `ISAM` when doing some `ORDER BY ... DESC` queries.
- Fixed bug when doing a join on a text key which didn't cover the whole key.
- Option `--delay-key-write` didn't enable delayed key writing.
- Fixed update of `TEXT` column which involved only case changes.

- Fixed that `INSERT DELAYED` doesn't update timestamps that are given.
- Added function `YEARWEEK()` and options `x`, `X`, `v` and `V` to `DATE_FORMAT()`.
- Fixed problem with `MAX(indexed_column)` and `HEAP` tables.
- Fixed problem with `BLOB NULL` keys and `LIKE "prefix%"`.
- Fixed problem with `MyISAM` and fixed-length rows < 5 bytes.
- Fixed problem that could cause MySQL to touch freed memory when doing very complicated `GROUP BY` queries.
- Fixed core dump if you got a crashed table where an `ENUM` field value was too big.

D.4.54 Alterações na distribuição 3.23.7 (10 Dec 1999)

- Fixed workaround under Linux to avoid problems with `pthread_mutex_timedwait`, which is used with `INSERT DELAYED`. Veja Seção 2.6.2 [Linux], Página 137.
- Fixed that one will get a 'disk full' error message if one gets disk full when doing sorting (instead of waiting until we got more disk space).
- Fixed a bug in `MyISAM` with keys > 250 characters.
- In `MyISAM` one can now do an `INSERT` at the same time as other threads are reading from the table.
- Added `max_write_lock_count` variable to `mysqld` to force a `READ` lock after a certain number of `WRITE` locks.
- Inverted flag `delay_key_write` on `show variables`.
- Renamed `concurrency` variable to `thread_concurrency`.
- The following functions are now multi-byte-safe: `LOCATE(substr,str)`, `POSITION(substr IN str)`, `LOCATE(substr,str,pos)`, `INSTR(str,substr)`, `LEFT(str,len)`, `RIGHT(str,len)`, `SUBSTRING(str,pos,len)`, `SUBSTRING(str FROM pos FOR len)`, `MID(str,pos,len)`, `SUBSTRING(str,pos)`, `SUBSTRING(str FROM pos)`, `SUBSTRING_INDEX(str,delim,count)`, `RTRIM(str)`, `TRIM([[BOTH | TRAILING] [remstr] FROM] str)`, `REPLACE(str,from_str,to_str)`, `REVERSE(str)`, `INSERT(str,pos,len,newstr)`, `LCASE(str)`, `LOWER(str)`, `UCASE(str)` and `UPPER(str)`; patch by Wei He.
- Fix core dump when releasing a lock from a non-existent table.
- Remove locks on tables before starting to remove duplicates.
- Added option `FULL` to `SHOW PROCESSLIST`.
- Added option `--verbose` to `mysqladmin`.
- Fixed problem when automatically converting `HEAP` to `MyISAM`.
- Fixed bug in `HEAP` tables when doing insert + delete + insert + scan the table.
- Fixed bugs on Alpha with `REPLACE()` and `LOAD DATA INFILE`.
- Added `interactive_timeout` variable to `mysqld`.
- Changed the argument to `mysql_data_seek()` from `ulong` to `ulonglong`.

D.4.55 Alterações na distribuição 3.23.6

- Added `-O lower_case_table_names={0|1}` option to `mysqld` to allow users to force table names to lowercase.
- Added `SELECT ... INTO DUMPFILE`.
- Added `--ansi` option to `mysqld` to make some functions SQL-99 compatible.
- Temporary table names now start with `#sql`.
- Added quoting of identifiers with ‘ (`"` in `--ansi` mode).
- Changed to use `snprintf()` when printing floats to avoid some buffer overflows on FreeBSD.
- Made `FLOOR()` overflow safe on FreeBSD.
- Added `--quote-names` option to `mysqldump`.
- Fixed bug that one could make a part of a `PRIMARY KEY NOT NULL`.
- Fixed `encrypt()` to be thread-safe and not reuse buffer.
- Added `mysql_odbc_escape_string()` function to support big5 characters in MyODBC.
- Rewrote the storage engine to use classes. This introduces a lot of new code, but will make table handling faster and better.
- Added patch by Sasha for user-defined variables.
- Changed that `FLOAT` and `DOUBLE` (without any length modifiers) no longer are fixed decimal point numbers.
- Changed the meaning of `FLOAT(X)`: Now this is the same as `FLOAT` if `X` <= 24 and a `DOUBLE` if 24 < `X` <= 53.
- `DECIMAL(X)` is now an alias for `DECIMAL(X,0)` and `DECIMAL` is now an alias for `DECIMAL(10,0)`. The same goes for `NUMERIC`.
- Added option `ROW_FORMAT={default | dynamic | fixed | compressed}` to `CREATE_TABLE`.
- `DELETE FROM table_name` didn’t work on temporary tables.
- Changed function `CHAR_LENGTH()` to be multi-byte character safe.
- Added function `ORD(string)`.

D.4.56 Alterações na distribuição 3.23.5 (20 Oct 1999)

- Fixed some Y2K problems in the new date handling in 3.23.
- Fixed problem with `SELECT DISTINCT ... ORDER BY RAND()`.
- Added patches by Sergei A. Golubchik for text searching on the `MyISAM` level.
- Fixed cache overflow problem when using full joins without keys.
- Fixed some configure issues.
- Some small changes to make parsing faster.
- Adding a column after the last field with `ALTER TABLE` didn’t work.
- Fixed problem when using an `AUTO_INCREMENT` column in two keys

- With `MyISAM`, you now can have an `AUTO_INCREMENT` column as a key sub part: `CREATE TABLE foo (a INT NOT NULL AUTO_INCREMENT, b CHAR(5), PRIMARY KEY (b,a))`
- Fixed bug in `MyISAM` with packed char keys that could be `NULL`.
- `AS` on field name with `CREATE TABLE table_name SELECT ...` didn't work.
- Allow use of `NATIONAL` and `NCHAR` when defining character columns. This is the same as not using `BINARY`.
- Don't allow `NULL` columns in a `PRIMARY KEY` (only in `UNIQUE` keys).
- Clear `LAST_INSERT_ID()` if one uses this in ODBC: `WHERE auto_increment_column IS NULL`. This seems to fix some problems with Access.
- `SET SQL_AUTO_IS_NULL=0|1` now turns on/off the handling of searching after the last inserted row with `WHERE auto_increment_column IS NULL`.
- Added new variable `concurrency` to `mysqld` for Solaris.
- Added `--relative` option to `mysqladmin` to make `extended-status` more useful to monitor changes.
- Fixed bug when using `COUNT(DISTINCT ...)` on an empty table.
- Added support for the Chinese character set GBK.
- Fixed problem with `LOAD DATA INFILE` and `BLOB` columns.
- Added bit operator `~` (negation).
- Fixed problem with `UDF` functions.

D.4.57 Alterações na distribuição 3.23.4 (28 Sep 1999)

- Inserting a `DATETIME` into a `TIME` column no longer will try to store 'days' in it.
- Fixed problem with storage of float/double on little endian machines. (This affected `SUM()`.)
- Added connect timeout on TCP/IP connections.
- Fixed problem with `LIKE "%"` on an index that may have `NULL` values.
- `REVOKE ALL PRIVILEGES` didn't revoke all privileges.
- Allow creation of temporary tables with same name as the original table.
- When granting a user a `GRANT` option for a database, he couldn't grant privileges to other users.
- New command: `SHOW GRANTS FOR user` (by Sinisa).
- New `date_add` syntax: `date/datetime + INTERVAL # interval_type`. By Joshua Chamas.
- Fixed privilege check for `LOAD DATA REPLACE`.
- Automatic fixing of broken include files on Solaris 2.7
- Some configure issues to fix problems with big filesystem detection.
- `REGEXP` is now case-insensitive if you use non-binary strings.

D.4.58 Alterações na distribuição 3.23.3

- Added patches for MIT-pthreads on NetBSD.
- Fixed range bug in `MyISAM`.
- `ASC` is now the default again for `ORDER BY`.
- Added `LIMIT` to `UPDATE`.
- Added `mysql_change_user()` function to the MySQL C API.
- Added character set to `SHOW VARIABLES`.
- Added support of `--[whitespace]` comments.
- Allow `INSERT into tbl_name VALUES ()`, that is, you may now specify an empty value list to insert a row in which each column is set to its default value.
- Changed `SUBSTRING(text FROM pos)` to conform to SQL-99. (Before this construct returned the rightmost `pos` characters.)
- `SUM()` with `GROUP BY` returned 0 on some systems.
- Changed output for `SHOW TABLE STATUS`.
- Added `DELAY_KEY_WRITE` option to `CREATE TABLE`.
- Allow `AUTO_INCREMENT` on any key part.
- Fixed problem with `YEAR(NOW())` and `YEAR(CURDATE())`.
- Added `CASE` construct.
- New function `COALESCE()`.

D.4.59 Alterações na distribuição 3.23.2 (09 Aug 1999)

- Fixed range optimiser bug: `SELECT * FROM table_name WHERE key_part1 >= const AND (key_part2 = const OR key_part2 = const)`. The bug was that some rows could be duplicated in the result.
- Running `myisamchk` without `-a` updated the index distribution incorrectly.
- `SET SQL_LOW_PRIORITY_UPDATES=1` was causing a parse error.
- You can now update index columns that are used in the `WHERE` clause. `UPDATE tbl_name SET KEY=KEY+1 WHERE KEY > 100`
- Date handling should now be a bit faster.
- Added handling of fuzzy dates (dates where day or month is 0), such as `'1999-01-00'`.
- Fixed optimization of `SELECT ... WHERE key_part1=const1 AND key_part_2=const2 AND key_part1=const4 AND key_part2=const4`; indextype should be `range` instead of `ref`.
- Fixed `egcs` 1.1.2 optimiser bug (when using `BLOB` values) on Linux Alpha.
- Fixed problem with `LOCK TABLES` combined with `DELETE FROM table`.
- `MyISAM` tables now allow keys on `NULL` and `BLOB/TEXT` columns.
- The following join is now much faster: `SELECT ... FROM t1 LEFT JOIN t2 ON ... WHERE t2.not_null_column IS NULL`.
- `ORDER BY` and `GROUP BY` can be done on functions.

- Changed handling of 'const_item' to allow handling of `ORDER BY RAND()`.
- Indexes are now used for `WHERE key_column = function`.
- Indexes are now used for `WHERE key_column = col_name` even if the columns are not identically packed.
- Indexes are now used for `WHERE col_name IS NULL`.
- Changed heap tables to be stored in low_byte_first order (to make it easy to convert to `MyISAM` tables)
- Automatic change of `HEAP` temporary tables to `MyISAM` tables in case of "table is full" errors.
- Added `--init-file=file_name` option to `mysqld`.
- Added `COUNT(DISTINCT value, [value, ...])`.
- `CREATE TEMPORARY TABLE` now creates a temporary table, in its own namespace, that is automatically deleted if connection is dropped.
- New reserved words (required for `CASE`): `CASE, THEN, WHEN, ELSE and END`.
- New functions `EXPORT_SET()` and `MD5()`.
- Support for the GB2312 Chinese character set.

D.4.60 Alterações na distribuição 3.23.1

- Fixed some compilation problems.

D.4.61 Alterações na distribuição 3.23.0 (05 Aug 1999: Alpha)

- A new storage engine library (`MyISAM`) with a lot of new features. Veja Seção 7.1 [MyISAM], Página 631.
- You can create in-memory `HEAP` tables which are extremely fast for lookups.
- Support for big files (63-bit) on OSs that support big files.
- New function `LOAD_FILE(filename)` to get the contents of a file as a string value.
- New operator `<=>` which will act as `=` but will return TRUE if both arguments are `NULL`. This is useful for comparing changes between tables.
- Added the ODBC 3.0 `EXTRACT(interval FROM datetime)` function.
- Columns defined as `FLOAT(X)` are not rounded on storage and may be in scientific notation (1.0 E+10) when retrieved.
- `REPLACE` is now faster than before.
- Changed `LIKE` character comparison to behave as `=`; This means that `'e' LIKE 'é'` is now true. (If the line doesn't display correctly, the latter 'e' is a French 'e' with a dot above.)
- `SHOW TABLE STATUS` returns a lot of information about the tables.
- Added `LIKE` to the `SHOW STATUS` command.
- Added `Privileges` column to `SHOW COLUMNS`.
- Added `Packed` and `Comment` columns to `SHOW INDEX`.
- Added comments to tables (with `CREATE TABLE ... COMMENT "xxx"`).

- Added `UNIQUE`, as in `CREATE TABLE table_name (col INT not null UNIQUE)`
- New create syntax: `CREATE TABLE table_name SELECT ...`
- New create syntax: `CREATE TABLE IF NOT EXISTS ...`
- Allow creation of `CHAR(0)` columns.
- `DATE_FORMAT()` now requires '`%`' before any format character.
- `DELAYED` is now a reserved word (sorry about that :( ).
- An example procedure is added: `analyse`, file: '`sql_analyse.c`'. This will describe the data in your query. Try the following:

```
SELECT ... FROM ...
WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
```

  This procedure is extremely useful when you want to check the data in your table!
- `BINARY` cast to force a string to be compared in case-sensitive fashion.
- Added `--skip-show-database` option to `mysqld`.
- Check whether a row has changed in an `UPDATE` now also works with `BLOB`/`TEXT` columns.
- Added the `INNER` join syntax. **NOTE**: This made `INNER` a reserved word!
- Added support for netmasks to the hostname in the MySQL grant tables. You can specify a netmask using the `IP/NETMASK` syntax.
- If you compare a `NOT NULL DATE/DATETIME` column with `IS NULL`, this is changed to a compare against `0` to satisfy some ODBC applications. (By `shreeve@uci.edu`.)
- `NULL IN (...)` now returns `NULL` instead of `0`. This will ensure that `null_column NOT IN (...)` doesn't match `NULL` values.
- Fix storage of floating-point values in `TIME` columns.
- Changed parsing of `TIME` strings to be more strict. Now the fractional second part is detected (and currently skipped). The following formats are supported:
  - [[DAYS] [H]H:]MM:]SS[.fraction]
  - [[[[H]H]H]H]MM]SS[.fraction]
- Detect (and ignore) fractional second part from `DATETIME`.
- Added the `LOW_PRIORITY` attribute to `LOAD DATA INFILE`.
- The default index name now uses the same case as the column name on which the index name is based.
- Changed default number of connections to 100.
- Use bigger buffers when using `LOAD DATA INFILE`.
- `DECIMAL(x,y)` now works according to SQL-99.
- Added aggregate UDF functions. Thanks to Andreas F. Bobak (`bobak@relog.ch`) for this!
- `LAST_INSERT_ID()` is now updated for `INSERT INTO ... SELECT`.
- Some small changes to the join table optimiser to make some joins faster.
- `SELECT DISTINCT` is much faster; it uses the new `UNIQUE` functionality in `MyISAM`. One difference compared to MySQL Version 3.22 is that the output of `DISTINCT` is no longer sorted.

- All C client API macros are now functions to make shared libraries more reliable. Because of this, you can no longer call `mysql_num_fields()` on a `MYSQL` object, you must use `mysql_field_count()` instead.
- Added use of `LIBWRAP`; patch by Henning P. Schmiedehausen.
- Don't allow `AUTO_INCREMENT` for other than numerical columns.
- Using `AUTO_INCREMENT` will now automatically make the column `NOT NULL`.
- Show `NULL` as the default value for `AUTO_INCREMENT` columns.
- Added `SQL_BIG_RESULT`; `SQL_SMALL_RESULT` is now default.
- Added a shared library RPM. This enhancement was contributed by David Fox (`dsfox@cogsci.ucsd.edu`).
- Added `--enable-large-files` and `--disable-large-files` switches to `configure`. See '`configure.in`' for some systems where this is automatically turned off because of broken implementations.
- Upgraded `readline` to 4.0.
- New `CREATE TABLE` options: `PACK_KEYS` and `CHECKSUM`.
- Added `--default-table-type` option to `mysqld`.

D.5 Alterações na distribuição 3.22.x (Old; discontinued)

The 3.22 version has faster and safer connect code than version 3.21, as well as a lot of new nice enhancements. As there aren't really any major changes, upgrading from 3.21 to 3.22 should be very easy and painless. Veja Seção 2.5.4 [Upgrading-from-3.21], Página 129.

D.5.1 Alterações na distribuição 3.22.35

- Fixed problem with `STD()`.
- Merged changes from the newest `ISAM` library from 3.23.
- Fixed problem with `INSERT DELAYED`.
- Fixed a bug core dump when using a `LEFT JOIN/STRAIGHT_JOIN` on a table with only one row.

D.5.2 Alterações na distribuição 3.22.34

- Fixed problem with `GROUP BY` on `TINYBLOB` columns; this caused bugzilla to not show rows in some queries.
- Had to do total recompile of the Windows binary version as VC++ didn't compile all relevant files for 3.22.33 :(

D.5.3 Alterações na distribuição 3.22.33

- Fixed problems in Windows when locking tables with `LOCK TABLE`.
- Quicker kill of `SELECT DISTINCT` queries.

D.5.4 Alterações na distribuição 3.22.32 (14 Feb 2000)

- Fixed problem when storing numbers in timestamps.
- Fix problem with timezones that have half hour offsets.
- Added `mysqlhotcopy`, a fast online hot-backup utility for local MySQL databases. By Tim Bunce.
- New more secure `mysqlaccess`. Thanks to Steve Harvey for this.
- Fixed security problem in the protocol regarding password checking.
- Fixed problem that affected queries that did arithmetic on `GROUP` functions.
- Fixed a bug in the `ISAM` code when deleting rows on tables with packed indexes.

D.5.5 Alterações na distribuição 3.22.31

- A few small fixes for the Windows version.

D.5.6 Alterações na distribuição 3.22.30

- Fixed optimiser problem on `SELECT` when using many overlapping indexes.
- Disabled floating-point exceptions for FreeBSD to fix core dump when doing `SELECT FLOOR(POW(2,63))`.
- Added print of default arguments options to all clients.
- Fixed critical problem with the `WITH GRANT OPTION` option.
- Fixed non-critical Y2K problem when writing short date to log files.

D.5.7 Alterações na distribuição 3.22.29 (02 Jan 2000)

- Upgraded the configure and include files to match the latest 3.23 version. This should increase portability and make it easier to build shared libraries.
- Added latest patches to MIT-pthreads for NetBSD.
- Fixed problem with timezones that are < GMT -11.
- Fixed a bug when deleting packed keys in NISAM.
- Fixed problem that could cause MySQL to touch freed memory when doing very complicated `GROUP BY` queries.
- Fixed core dump if you got a crashed table where an `ENUM` field value was too big.
- Added `mysqlshutdown.exe` and `mysqlwatch.exe` to the Windows distribution.
- Fixed problem when doing `ORDER BY` on a reference key.
- Fixed that `INSERT DELAYED` doesn't update timestamps that are given.

D.5.8 Alterações na distribuição 3.22.28 (20 Oct 1999)

- Fixed problem with `LEFT JOIN` and `COUNT()` on a column which was declared `NULL +` and it had a `DEFAULT` value.
- Fixed core dump problem when using `CONCAT()` in a `WHERE` clause.
- Fixed problem with `AVG()` and `STD()` with `NULL` values.

D.5.9 Alterações na distribuição 3.22.27

- Fixed prototype in '`my_ctype.h`' when using other character sets.
- Some configure issues to fix problems with big filesystem detection.
- Fixed problem when sorting on big `BLOB` columns.
- `ROUND()` will now work on Windows.

D.5.10 Alterações na distribuição 3.22.26 (16 Sep 1999)

- Fixed core dump with empty `BLOB/TEXT` column argument to `REVERSE()`.
- Extended `/*! */` with version numbers.
- Changed `SUBSTRING(text FROM pos)` to conform to SQL-99. (Before this construct returned the rightmost 'pos' characters.)
- Fixed problem with `LOCK TABLES` combined with `DELETE FROM table`
- Fixed problem that `INSERT ... SELECT` didn't use `BIG_TABLES`.
- `SET SQL_LOW_PRIORITY_UPDATES=#` didn't work.
- Password wasn't updated correctly if privileges didn't change on: `GRANT ... IDENTIFIED BY`
- Fixed range optimiser bug in `SELECT * FROM table_name WHERE key_part1 >= const AND (key_part2 = const OR key_part2 = const)`.
- Fixed bug in compression key handling in `ISAM`.

D.5.11 Alterações na distribuição 3.22.25

- Fixed some small problems with the installation.

D.5.12 Alterações na distribuição 3.22.24 (05 Jul 1999)

- `DATA` is no longer a reserved word.
- Fixed optimiser bug with tables with only one row.
- Fixed bug when using `LOCK TABLES table_name READ; FLUSH TABLES;`
- Applied some patches for HP-UX.
- `isamchk` should now work on Windows.
- Changed '`configure`' to not use big file handling on Linux as this crashes some Red Hat 6.0 systems

D.5.13 Alterações na distribuição 3.22.23 (08 Jun 1999)

- Upgraded to use Autoconf 2.13, Automake 1.4 and `libtool` 1.3.2.
- Better support for SCO in `configure`.
- Added option `--defaults-file=file_name` to option file handling to force use of only one specific option file.
- Extended `CREATE` syntax to ignore MySQL Version 3.23 keywords.

- Fixed deadlock problem when using `INSERT DELAYED` on a table locked with `LOCK TABLES`.
- Fixed deadlock problem when using `DROP TABLE` on a table that was locked by another thread.
- Add logging of `GRANT/REVOKE` commands in the update log.
- Fixed `isamchk` to detect a new error condition.
- Fixed bug in `NATURAL LEFT JOIN`.

D.5.14 Alterações na distribuição 3.22.22 (30 Apr 1999)

- Fixed problem in the C API when you called `mysql_close()` directly after `mysql_init()`.
- Better client error message when you can't open socket.
- Fixed `delayed_insert_thread` counting when you couldn't create a new delayed_insert thread.
- Fixed bug in `CONCAT()` with many arguments.
- Added patches for DEC 3.2 and SCO.
- Fixed path-bug when installing MySQL as a service on NT.
- MySQL on Windows is now compiled with VC++ 6.0 instead of with VC++ 5.0.
- New installation setup for MySQL on Windows.

D.5.15 Alterações na distribuição 3.22.21

- Fixed problem with `DELETE FROM TABLE` when table was locked by another thread.
- Fixed bug in `LEFT JOIN` involving empty tables.
- Changed the `mysql.db` column from `CHAR(32)` to `CHAR(60)`.
- `MODIFY` and `DELAYED` are no longer reserved words.
- Fixed a bug when storing days in a `TIME` column.
- Fixed a problem with `Host '...' is not allowed to connect to this MySQL server` after one had inserted a new MySQL user with a `GRANT` command.
- Changed to use `TCP_NODELAY` also on Linux (should give faster TCP/IP connections).

D.5.16 Alterações na distribuição 3.22.20 (18 Mar 1999)

- Fixed `STD()` for big tables when result should be 0.
- The update log didn't have newlines on some operating systems.
- `INSERT DELAYED` had some garbage at end in the update log.

D.5.17 Alterações na distribuição 3.22.19 (Mar 1999: Production)

- Fixed bug in `mysql_install_db` (from 3.22.17).
- Changed default key cache size to 8M.
- Fixed problem with queries that needed temporary tables with `BLOB` columns.

D.5.18 Alterações na distribuição 3.22.18

- Fixes a fatal problem in 3.22.17 on Linux; after `shutdown` not all threads died properly.
- Added option `-O flush_time=#` to `mysqld`. This is mostly useful on Windows and tells how often MySQL should close all unused tables and flush all updated tables to disk.
- Fixed problem that a `VARCHAR` column compared with `CHAR` column didn't use keys efficiently.

D.5.19 Alterações na distribuição 3.22.17

- Fixed a core dump problem when using `--log-update` and connecting without a default database.
- Fixed some `configure` and portability problems.
- Using `LEFT JOIN` on tables that had circular dependencies caused `mysqld` to hang forever.

D.5.20 Alterações na distribuição 3.22.16 (Feb 1999: Gamma)

- `mysqladmin processlist` could kill the server if a new user logged in.
- `DELETE FROM tbl_name WHERE key_column=col_name` didn't find any matching rows. Fixed.
- `DATE_ADD(column, ...)` didn't work.
- `INSERT DELAYED` could deadlock with status 'upgrading lock'
- Extended `ENCRYPT()` to take longer salt strings than 2 characters.
- `longlong2str` is now much faster than before. For `Intel x86` platforms, this function is written in optimised assembler.
- Added the `MODIFY` keyword to `ALTER TABLE`.

D.5.21 Alterações na distribuição 3.22.15

- `GRANT` used with `IDENTIFIED BY` didn't take effect until privileges were flushed.
- Name change of some variables in `SHOW STATUS`.
- Fixed problem with `ORDER BY` with 'only index' optimization when there were multiple key definitions for a used column.
- `DATE` and `DATETIME` columns are now up to 5 times faster than before.
- `INSERT DELAYED` can be used to let the client do other things while the server inserts rows into a table.
- `LEFT JOIN USING (col1,col2)` didn't work if one used it with tables from 2 different databases.
- `LOAD DATA LOCAL INFILE` didn't work in the Unix version because of a missing file.
- Fixed problems with `VARCHAR/BLOB` on very short rows (< 4 bytes); error 127 could occur when deleting rows.
- Updating `BLOB/TEXT` through formulas didn't work for short (< 256 char) strings.

- When you did a `GRANT` on a new host, `mysqld` could die on the first connect from this host.
- Fixed bug when one used `ORDER BY` on column name that was the same name as an alias.
- Added `BENCHMARK(loop_count,expression)` function to time expressions.

D.5.22 Alterações na distribuição 3.22.14

- Allow empty arguments to `mysqld` to make it easier to start from shell scripts.
- Setting a `TIMESTAMP` column to `NULL` didn't record the timestamp value in the update log.
- Fixed lock handler bug when one did `INSERT INTO TABLE ... SELECT ... GROUP BY`.
- Added a patch for `localtime_r()` on Windows so that it will no lonher crash if your date is > 2039, but instead will return a time of all zero.
- Names for user-defined functions are no longer case-sensitive.
- Added escape of `^Z` (ASCII 26) to `\Z` as `^Z` doesn't work with pipes on Windows.
- `mysql_fix_privileges` adds a new column to the `mysql.func` to support aggregate UDF functions in future MySQL releases.

D.5.23 Alterações na distribuição 3.22.13

- Saving `NOW()`, `CURDATE()` or `CURTIME()` directly in a column didn't work.
- `SELECT COUNT(*) ... LEFT JOIN ...` didn't work with no `WHERE` part.
- Updated '`config.guess`' to allow MySQL to configure on UnixWare 7.1.x.
- Changed the implementation of `pthread_cond()` on the Windows version. `get_lock()` now correctly times out on Windows!

D.5.24 Alterações na distribuição 3.22.12

- Fixed problem when using `DATE_ADD()` and `DATE_SUB()` in a `WHERE` clause.
- You can now set the password for a user with the `GRANT ... TO user IDENTIFIED BY 'password'` syntax.
- Fixed bug in `GRANT` checking with `SELECT` on many tables.
- Added missing file `mysql_fix_privilege_tables` to the RPM distribution. This is not run by default because it relies on the client package.
- Added option `SQL_SMALL_RESULT` to `SELECT` to force use of fast temporary tables when you know that the result set will be small.
- Allow use of negative real numbers without a decimal point.
- Day number is now adjusted to maximum days in month if the resulting month after `DATE_ADD/DATE_SUB()` doesn't have enough days.
- Fix that `GRANT` compares columns in case-insensitive fashion.
- Fixed a bug in '`sql_list.h`' that made `ALTER TABLE` dump core in some contexts.

- The hostname in `user@hostname` can now include '`.`' and '`-`' without quotes in the context of the `GRANT`, `REVOKE` and `SET PASSWORD FOR ...` statements.
- Fix for `isamchk` for tables which need big temporary files.

D.5.25 Alterações na distribuição 3.22.11

- **Important**: You must run the `mysql_fix_privilege_tables` script when you upgrade to this version! This is needed because of the new `GRANT` system. If you don't do this, you will get `Access denied` when you try to use `ALTER TABLE`, `CREATE INDEX`, or `DROP INDEX`.
- `GRANT` to allow/deny users table and column access.
- Changed `USER()` to return a value in `user@host` format. Formerly it returned only `user`.
- Changed the syntax for how to set `PASSWORD` for another user.
- New command `FLUSH STATUS` that resets most status variables to zero.
- New status variables: `aborted_threads`, `aborted_connects`.
- New option variable: `connection_timeout`.
- Added support for Thai sorting (by Pruet Boonma `pruet@ds90.intanon.nectec.or.th`).
- Slovak and Japanese error messages.
- Configuration and portability fixes.
- Added option `SET SQL_WARNINGS=1` to get a warning count also for simple (single-row) inserts.
- MySQL now uses `SIGTERM` instead of `SIGQUIT` with shutdown to work better on FreeBSD.
- Added option `\G` (print vertically) to `mysql`.
- `SELECT HIGH_PRIORITY ...` killed `mysqld`.
- `IS NULL` on a `AUTO_INCREMENT` column in a `LEFT JOIN` didn't work as expected.
- New function `MAKE_SET()`.

D.5.26 Alterações na distribuição 3.22.10

- `mysql_install_db` no longer starts the MySQL server! You should start `mysqld` with `safe_mysqld` after installing it! The MySQL RPM will, however, start the server as before.
- Added `--bootstrap` option to `mysqld` and recoded `mysql_install_db` to use it. This will make it easier to install MySQL with RPMs.
- Changed `+`, `-` (sign and minus), `*`, `/`, `%`, `ABS()` and `MOD()` to be `BIGINT` aware (64-bit safe).
- Fixed a bug in `ALTER TABLE` that caused `mysqld` to crash.
- MySQL now always reports the conflicting key values when a duplicate key entry occurs. (Before this was only reported for `INSERT`.)
- New syntax: `INSERT INTO tbl_name SET col_name=value, col_name=value, ...`

- Most errors in the '`.err`' log are now prefixed with a time stamp.
- Added option `MYSQL_INIT_COMMAND` to `mysql_options()` to make a query on connect or reconnect.
- Added option `MYSQL_READ_DEFAULT_FILE` and `MYSQL_READ_DEFAULT_GROUP` to `mysql_options()` to read the following parameters from the MySQL option files: `port`, `socket`, `compress`, `password`, `pipe`, `timeout`, `user`, `init-command`, `host` and `database`.
- Added `maybe_null` to the UDF structure.
- Added option `IGNORE` to `INSERT` statements with many rows.
- Fixed some problems with sorting of the `koi8` character sets; users of `koi8` **must** run `isamchk -rq` on each table that has an index on a `CHAR` or `VARCHAR` column.
- New script `mysql_setpermission`, by Luuk de Boer. It allows easy creation of new users with permissions for specific databases.
- Allow use of hexadecimal strings (0x...) when specifying a constant string (like in the column separators with `LOAD DATA INFILE`).
- Ported to OS/2 (thanks to Antony T. Curtis `antony.curtis@olcs.net`).
- Added more variables to `SHOW STATUS` and changed format of output to be like `SHOW VARIABLES`.
- Added `extended-status` command to `mysqladmin` which will show the new status variables.

D.5.27 Alterações na distribuição 3.22.9

- `SET SQL_LOG_UPDATE=0` caused a lockup of the server.
- New SQL command: `FLUSH [ TABLES | HOSTS | LOGS | PRIVILEGES ] [, ...]`
- New SQL command: `KILL thread_id`.
- Added casts and changed include files to make MySQL easier to compile on AIX and DEC OSF/1 4.x
- Fixed conversion problem when using `ALTER TABLE` from a `INT` to a short `CHAR()` column.
- Added `SELECT HIGH_PRIORITY`; this will get a lock for the `SELECT` even if there is a thread waiting for another `SELECT` to get a `WRITE LOCK`.
- Moved `wild_compare()` to string class to be able to use `LIKE` on `BLOB/TEXT` columns with `\0`.
- Added `ESCAPE` option to `LIKE`.
- Added a lot more output to `mysqladmin debug`.
- You can now start `mysqld` on Windows with the `--flush` option. This will flush all tables to disk after each update. This makes things much safer on the Windows platforms but also **much** slower.

D.5.28 Alterações na distribuição 3.22.8

- Czech character sets should now work much better.

- `DATE_ADD()` and `DATE_SUB()` didn't work with group functions.
- `mysql` will now also try to reconnect on `USE database` commands.
- Fix problem with `ORDER BY` and `LEFT JOIN` and `const` tables.
- Fixed problem with `ORDER BY` if the first `ORDER BY` column was a key and the rest of the `ORDER BY` columns wasn't part of the key.
- Fixed a big problem with `OPTIMIZE TABLE`.
- MySQL clients on NT will now by default first try to connect with named pipes and after this with TCP/IP.
- Fixed a problem with `DROP TABLE` and `mysqladmin shutdown` on Windows (a fatal bug from 3.22.6).
- Fixed problems with `TIME columns` and negative strings.
- Added an extra thread signal loop on shutdown to avoid some error messages from the client.
- MySQL now uses the next available number as extension for the update log file.
- Added patches for UNIXWARE 7.

D.5.29 Alterações na distribuição 3.22.7 (Sep 1998: Beta)

- Added `LIMIT` clause for the `DELETE` statement.
- You can now use the `/*! ... */` syntax to hide MySQL-specific keywords when you write portable code. MySQL will parse the code inside the comments as if the surrounding `/*!` and `*/` comment characters didn't exist.
- `OPTIMIZE TABLE tbl_name` can now be used to reclaim disk space after many deletes. Currently, this uses `ALTER TABLE` to regenerate the table, but in the future it will use an integrated `isamchk` for more speed.
- Upgraded `libtool` to get the configure more portable.
- Fixed slow `UPDATE` and `DELETE` operations when using `DATETIME` or `DATE` keys.
- Changed optimiser to make it better at deciding when to do a full join and when using keys.
- You can now use `mysqladmin proc` to display information about your own threads. Only users with the `PROCESS` privilege can get information about all threads. (In 4.0.2 one needs the `SUPER` privilege for this.)
- Added handling of formats `YYMMDD`, `YYYYMMDD`, `YYMMDDHHMMSS` for numbers when using `DATETIME` and `TIMESTAMP` types. (Formerly these formats only worked with strings.)
- Added connect option `CLIENT_IGNORE_SPACE` to allow use of spaces after function names and before '`(`' (Powerbuilder requires this). This will make all function names reserved words.
- Added the `--log-long-format` option to `mysqld` to enable timestamps and INSERT_IDs in the update log.
- Added `--where` option to `mysqldump` (patch by Jim Faucette).
- The lexical analyser now uses "perfect hashing" for faster parsing of SQL statements.

D.5.30 Alterações na distribuição 3.22.6

- Faster `mysqldump`.
- For the `LOAD DATA INFILE` statement, you can now use the new `LOCAL` keyword to read the file from the client. `mysqlimport` will automatically use `LOCAL` when importing with the TCP/IP protocol.
- Fixed small optimise problem when updating keys.
- Changed makefiles to support shared libraries.
- MySQL-NT can now use named pipes, which means that you can now use MySQL-NT without having to install TCP/IP.

D.5.31 Alterações na distribuição 3.22.5

- All table lock handing is changed to avoid some very subtle deadlocks when using `DROP TABLE`, `ALTER TABLE`, `DELETE FROM TABLE` and `mysqladmin flush-tables` under heavy usage. Changed locking code to get better handling of locks of different types.
- Updated `DBI` to 1.00 and `DBD` to 1.2.0.
- Added a check that the error message file contains error messages suitable for the current version of `mysqld`. (To avoid errors if you accidentally try to use an old error message file.)
- All count structures in the client (`affected_rows()`, `insert_id()`, ...) are now of type `BIGINT` to allow 64-bit values to be used. This required a minor change in the MySQL protocol which should affect only old clients when using tables with `AUTO_INCREMENT` values > 16M.
- The return type of `mysql_fetch_lengths()` has changed from `uint *` to `ulong *`. This may give a warning for old clients but should work on most machines.
- Change `mysys` and `dbug` libraries to allocate all thread variables in one struct. This makes it easier to make a threaded '`libmysql.dll`' library.
- Use the result from `gethostname()` (instead of `uname()`) when constructing '`.pid`' file names.
- New better compressed server/client protocol.
- `COUNT()`, `STD()` and `AVG()` are extended to handle more than 4G rows.
- You can now store values in the range `-838:59:59 <= x <= 838:59:59` in a `TIME` column.
- **Warning: incompatible change!!** If you set a `TIME` column to too short a value, MySQL now assumes the value is given as: `[[[D ]HH:]MM:]SS` instead of `HH[:MM[:SS]]`.
- `TIME_TO_SEC()` and `SEC_TO_TIME()` can now handle negative times and hours up to 32767.
- Added new option `SET SQL_LOG_UPDATE={0|1}` to allow users with the `PROCESS` privilege to bypass the update log. (Modified patch from Sergey A Mukhin `violet@rosnet.net`.)
- Fixed fatal bug in `LPAD()`.
- Initialise line buffer in '`mysql.cc`' to make `BLOB` reading from pipes safer.

- Added `-O max_connect_errors=#` option to `mysqld`. Connect errors are now reset for each correct connection.
- Increased the default value of `max_allowed_packet` to `1M` in `mysqld`.
- Added `--low-priority-updates` option to `mysqld`, to give table-modifying operations (`INSERT`, `REPLACE`, `UPDATE`, `DELETE`) lower priority than retrievals. You can now use `{INSERT | REPLACE | UPDATE | DELETE} LOW_PRIORITY ...` You can also use `SET SQL_LOW_PRIORITY_UPDATES={0|1}` to change the priority for one thread. One side effect is that `LOW_PRIORITY` is now a reserved word. :(
- Add support for `INSERT INTO table ... VALUES(...),(...),(...)`, to allow inserting multiple rows with a single statement.
- `INSERT INTO tbl_name` is now also cached when used with `LOCK TABLES`. (Previously only `INSERT ... SELECT` and `LOAD DATA INFILE` were cached.)
- Allow `GROUP BY` functions with `HAVING`:

```
mysql> SELECT col FROM table GROUP BY col HAVING COUNT(*)>0;
```

- `mysqld` will now ignore trailing '`;`' characters in queries. This is to make it easier to migrate from some other SQL servers that require the trailing '`;`'.
- Fix for corrupted fixed-format output generated by `SELECT INTO OUTFILE`.
- **Warning: incompatible change!** Added Oracle `GREATEST()` and `LEAST()` functions. You must now use these instead of the `MAX()` and `MIN()` functions to get the largest/smallest value from a list of values. These can now handle `REAL`, `BIGINT` and string (`CHAR` or `VARCHAR`) values.
- **Warning: incompatible change!** `DAYOFWEEK()` had offset 0 for Sunday. Changed the offset to 1.
- Give an error for queries that mix `GROUP BY` columns and fields when there is no `GROUP BY` specification.
- Added `--vertical` option to `mysql`, for printing results in vertical mode.
- Index-only optimization; some queries are now resolved using only indexes. Until MySQL 4.0, this works only for numeric columns. Veja Seção 5.4.3 [MySQL indexes], Página 447.
- Lots of new benchmarks.
- A new C API chapter and lots of other improvements in the manual.

D.5.32 Alterações na distribuição 3.22.4

- Added `--tmpdir` option to `mysqld`, for specifying the location of the temporary file directory.
- MySQL now automatically changes a query from an ODBC client:

```
SELECT ... FROM table WHERE auto_increment_column IS NULL
```

  to:

```
SELECT ... FROM table WHERE auto_increment_column == LAST_INSERT_ID()
```

  This allows some ODBC programs (Delphi, Access) to retrieve the newly inserted row to fetch the `AUTO_INCREMENT` id.

- `DROP TABLE` now waits for all users to free a table before deleting it.
- Fixed small memory leak in the new connect protocol.
- New functions `BIN()`, `OCT()`, `HEX()` and `CONV()` for converting between different number bases.
- Added function `SUBSTRING()` with 2 arguments.
- If you created a table with a record length smaller than 5, you couldn't delete rows from the table.
- Added optimization to remove `const` reference tables from `ORDER BY` and `GROUP BY`.
- `mysqld` now automatically disables system locking on Linux and Windows, and for systems that use MIT-pthreads. You can force the use of locking with the `--enable-external-locking` option.
- Added `--console` option to `mysqld`, to force a console window (for error messages) when using Windows.
- Fixed table locks for Windows.
- Allow '`$`' in identifiers.
- Changed name of user-specific configuration file from '`my.cnf`' to '`.my.cnf`' (Unix only).
- Added `DATE_ADD()` and `DATE_SUB()` functions.

D.5.33 Alterações na distribuição 3.22.3

- Fixed a lock problem (bug in MySQL Version 3.22.1) when closing temporary tables.
- Added missing `mysql_ping()` to the client library.
- Added `--compress` option to all MySQL clients.
- Changed `byte` to `char` in '`mysql.h`' and '`mysql_com.h`'.

D.5.34 Alterações na distribuição 3.22.2

- Searching on multiple constant keys that matched more than 30% of the rows didn't always use the best possible key.
- New functions `<<`, `>>`, `RPAD()` and `LPAD()`.
- You can now save default options (like passwords) in a configuration file ('`my.cnf`').
- Lots of small changes to get `ORDER BY` to work when no records are found when using fields that are not in `GROUP BY` (MySQL extension).
- Added `--chroot` option to `mysqld`, to start `mysqld` in a chroot environment (by Nikki Chumakov `nikkic@cityline.ru`).
- Trailing spaces are now ignored when comparing case-sensitive strings; this should fix some problems with ODBC and flag 512!
- Fixed a core dump bug in the range optimiser.
- Added `--one-thread` option to `mysqld`, for debugging with LinuxThreads (or `glibc`). (This replaces the `-T32` flag)
- Added `DROP TABLE IF EXISTS` to prevent an error from occurring if the table doesn't exist.

- `IF` and `EXISTS` are now reserved words (they would have to be sooner or later).
- Added lots of new options to `mysqldump`.
- Server error messages are now in '`mysqld_error.h`'.
- The server/client protocol now supports compression.
- All bug fixes from MySQL Version 3.21.32.

D.5.35 Alterações na distribuição 3.22.1 (Jun 1998: Alpha)

- Added new C API function `mysql_ping()`.
- Added new API functions `mysql_init()` and `mysql_options()`. You now MUST call `mysql_init()` before you call `mysql_real_connect()`. You don't have to call `mysql_init()` if you only use `mysql_connect()`.
- Added `mysql_options(...,MYSQL_OPT_CONNECT_TIMEOUT,...)` so you can set a timeout for connecting to a server.
- Added `--timeout` option to `mysqladmin`, as a test of `mysql_options()`.
- Added `AFTER column` and `FIRST` options to `ALTER TABLE ... ADD columns`. This makes it possible to add a new column at some specific location within a row in an existing table.
- `WEEK()` now takes an optional argument to allow handling of weeks when the week starts on Monday (some European countries). By default, `WEEK()` assumes the week starts on Sunday.
- `TIME` columns weren't stored properly (bug in MySQL Version 3.22.0).
- `UPDATE` now returns information about how many rows were matched and updated, and how many "warnings" occurred when doing the update.
- Fixed incorrect result from `FORMAT(-100,2)`.
- `ENUM` and `SET` columns were compared in binary (case-sensitive) fashion; changed to be case-insensitive.

D.5.36 Alterações na distribuição 3.22.0

- New (backward-compatible) connect protocol that allows you to specify the database to use when connecting, to get much faster connections to a specific database.

  The `mysql_real_connect()` call is changed to:

  ```
  mysql_real_connect(MYSQL *mysql, const char *host, const char *user,
                     const char *passwd, const char *db, uint port,
                     const char *unix_socket, uint client_flag)
  ```

- Each connection is handled by its own thread, rather than by the master `accept()` thread. This fixes permanently the telnet bug that was a topic on the mail list some time ago.
- All TCP/IP connections are now checked with backward-resolution of the hostname to get better security. `mysqld` now has a local hostname resolver cache so connections should actually be faster than before, even with this feature.

- A site automatically will be blocked from future connections if someone repeatedly connects with an "improper header" (like when one uses telnet).
- You can now refer to tables in different databases with references of the form `tbl_name@db_name` or `db_name.tbl_name`. This makes it possible to give a user read access to some tables and write access to others simply by keeping them in different databases!
- Added `--user` option to `mysqld`, to allow it to run as another Unix user (if it is started as the Unix `root` user).
- Added caching of users and access rights (for faster access rights checking)
- Normal users (not anonymous ones) can change their password with `mysqladmin password 'new_password'`. This uses encrypted passwords that are not logged in the normal MySQL log!
- All important string functions are now coded in assembler for x86 Linux machines. This gives a speedup of 10% in many cases.
- For tables that have many columns, the column names are now hashed for much faster column name lookup (this will speed up some benchmark tests a lot!)
- Some benchmarks are changed to get better individual timing. (Some loops were so short that a specific test took < 2 seconds. The loops have been changed to take about 20 seconds to make it easier to compare different databases. A test that took 1-2 seconds before now takes 11-24 seconds, which is much better)
- Re-arranged `SELECT` code to handle some very specific queries involving group functions (like `COUNT(*)`) without a `GROUP BY` but with `HAVING`. The following now works:

```
mysql> SELECT COUNT(*) as C FROM table HAVING C > 1;
```

- Changed the protocol for field functions to be faster and avoid some calls to `malloc()`.
- Added `-T32` option to `mysqld`, for running all queries under the main thread. This makes it possible to debug `mysqld` under Linux with `gdb`!
- Added optimization of `not_null_column IS NULL` (needed for some Access queries).
- Allow `STRAIGHT_JOIN` to be used between two tables to force the optimiser to join them in a specific order.
- String functions now return `VARCHAR` rather than `CHAR` and the column type is now `VARCHAR` for fields saved as `VARCHAR`. This should make the `MyODBC` driver better, but may break some old MySQL clients that don't handle `FIELD_TYPE_VARCHAR` the same way as `FIELD_TYPE_CHAR`.
- `CREATE INDEX` and `DROP INDEX` are now implemented through `ALTER TABLE`. `CREATE TABLE` is still the recommended (fast) way to create indexes.
- Added `--set-variable` option `wait_timeout` to `mysqld`.
- Added time column to `mysqladmin processlist` to show how long a query has taken or how long a thread has slept.
- Added lots of new variables to `show variables` and some new to `show status`.
- Added new type `YEAR`. `YEAR` is stored in 1 byte with allowable values of 0, and 1901 to 2155.
- Added new `DATE` type that is stored in 3 bytes rather than 4 bytes. All new tables are created with the new date type if you don't use the `--old-protocol` option to `mysqld`.

- Fixed bug in record caches; for some queries, you could get `Error from table handler: #` on some operating systems.
- Added `--enable-assembler` option to `configure`, for x86 machines (tested on Linux + `gcc`). This will enable assembler functions for the most important string functions for more speed!

D.6 Alterações na distribuição 3.21.x

Version 3.21 is quite old now, and should be avoided if possible. This information is kept here for historical purposes only.

D.6.1 Alterações na distribuição 3.21.33

- Fixed problem when sending `SIGHUP` to `mysqld`; `mysqld` core dumped when starting from boot on some systems.
- Fixed problem with losing a little memory for some connections.
- `DELETE FROM tbl_name` without a `WHERE` condition is now done the long way when you use `LOCK TABLES` or if the table is in use, to avoid race conditions.
- `INSERT INTO TABLE (timestamp_column) VALUES (NULL);` didn't set timestamp.

D.6.2 Alterações na distribuição 3.21.32

- Fixed some possible race conditions when doing many reopen/close on the same tables under heavy load! This can happen if you execute `mysqladmin refresh` often. This could in some very rare cases corrupt the header of the index file and cause error 126 or 138.
- Fixed fatal bug in `refresh()` when running with the `--skip-external-locking` option. There was a "very small" time gap after a `mysqladmin refresh` when a table could be corrupted if one thread updated a table while another thread did `mysqladmin refresh` and another thread started a new update ont the same table before the first thread had finished. A refresh (or `--flush-tables`) will now not return until all used tables are closed!
- `SELECT DISTINCT` with a `WHERE` clause that didn't match any rows returned a row in some contexts (bug only in 3.21.31).
- `GROUP BY + ORDER BY` returned one empty row when no rows where found.
- Fixed a bug in the range optimiser that wrote `Use_count: Wrong count for ...` in the error log file.

D.6.3 Alterações na distribuição 3.21.31

- Fixed a sign extension problem for the `TINYINT` type on Irix.
- Fixed problem with `LEFT("constant_string",function)`.
- Fixed problem with `FIND_IN_SET()`.

- `LEFT JOIN` core dumped if the second table is used with a constant `WHERE/ON` expression that uniquely identifies one record.
- Fixed problems with `DATE_FORMAT()` and incorrect dates. `DATE_FORMAT()` now ignores `'%'` to make it possible to extend it more easily in the future.

D.6.4 Alterações na distribuição 3.21.30

- `mysql` now returns an exit code `>` 0 if the query returned an error.
- Saving of command-line history to file in `mysql` client. By Tommy Larsen `tommy@mix.hive.no`.
- Fixed problem with empty lines that were ignored in '`mysql.cc`'.
- Save the pid of the signal handler thread in the pid file instead of the pid of the main thread.
- Added patch by `tommy@valley.ne.jp` to support Japanese characters SJIS and UJIS.
- Changed `safe_mysqld` to redirect startup messages to `'hostname'.err` instead of `'hostname'.log` to reclaim file space on `mysqladmin refresh`.
- `ENUM` always had the first entry as default value.
- `ALTER TABLE` wrote two entries to the update log.
- `sql_acc()` now closes the `mysql` grant tables after a reload to save table space and memory.
- Changed `LOAD DATA` to use less memory with tables and `BLOB` columns.
- Sorting on a function which made a division / 0 produced a wrong set in some cases.
- Fixed `SELECT` problem with `LEFT()` when using the `czech` character set.
- Fixed problem in `isamchk`; it couldn't repair a packed table in a very unusual case.
- `SELECT` statements with `&` or `|` (bit functions) failed on columns with `NULL` values.
- When comparing a field = field, where one of the fields was a part key, only the length of the part key was compared.

D.6.5 Alterações na distribuição 3.21.29

- `LOCK TABLES + DELETE from tbl_name` never removed locks properly.
- Fixed problem when grouping on an `OR` function.
- Fixed permission problem with `umask()` and creating new databases.
- Fixed permission problem on result file with `SELECT ... INTO OUTFILE ...`
- Fixed problem in range optimiser (core dump) for a very complex query.
- Fixed problem when using `MIN(integer)` or `MAX(integer)` in `GROUP BY`.
- Fixed bug on Alpha when using integer keys. (Other keys worked on Alpha.)
- Fixed bug in `WEEK("XXXX-xx-01")`.

D.6.6 Alterações na distribuição 3.21.28

- Fixed socket permission (clients couldn't connect to Unix socket on Linux).
- Fixed bug in record caches; for some queries, you could get `Error from table handler: #` on some operating systems.

D.6.7 Alterações na distribuição 3.21.27

- Added user level lock functions `GET_LOCK(string,timeout)`, `RELEASE_LOCK(string)`.
- Added `Opened_tables` to `show status`.
- Changed connect timeout to 3 seconds to make it somewhat harder for crackers to kill `mysqld` through telnet + TCP/IP.
- Fixed bug in range optimiser when using `WHERE key_part_1 >= something AND key_part_2 <= something_else`.
- Changed `configure` for detection of FreeBSD 3.0 9803xx and above
- `WHERE` with `string_col_key = constant_string` didn't always find all rows if the column had many values differing only with characters of the same sort value (like e and e with an accent).
- Strings keys looked up with 'ref' were not compared in case-sensitive fashion.
- Added `umask()` to make log files non-readable for normal users.
- Ignore users with old (8-byte) password on startup if not using `--old-protocol` option to `mysqld`.
- `SELECT` which matched all key fields returned the values in the case of the matched values, not of the found values. (Minor problem.)

D.6.8 Alterações na distribuição 3.21.26

- `FROM_DAYS(0)` now returns `"0000-00-00"`.
- In `DATE_FORMAT()`, PM and AM were swapped for hours 00 and 12.
- Extended the default maximum key size to 256.
- Fixed bug when using `BLOB/TEXT` in `GROUP BY` with many tables.
- An `ENUM` field that is not declared `NOT NULL` has `NULL` as the default value. (Previously, the default value was the first enumeration value.)
- Fixed bug in the join optimiser code when using many part keys on the same key: `INDEX (Organisation,Surname(35),Initials(35))`.
- Added some tests to the table order optimiser to get some cases with `SELECT ... FROM many_tables` much faster.
- Added a retry loop around `accept()` to possibly fix some problems on some Linux machines.

D.6.9 Alterações na distribuição 3.21.25

- Changed `typedef 'string'` to `typedef 'my_string'` for better portability.

- You can now kill threads that are waiting on a disk-full condition.
- Fixed some problems with UDF functions.
- Added long options to `isamchk`. Try `isamchk --help`.
- Fixed a bug when using 8 bytes long (alpha); `filesort()` didn't work. Affects `DISTINCT`, `ORDER BY` and `GROUP BY` on 64-bit processors.

D.6.10 Alterações na distribuição 3.21.24

- Dynamic loadable functions. Based on source from Alexis Mikhailov.
- You couldn't delete from a table if no one had done a `SELECT` on the table.
- Fixed problem with range optimiser with many `OR` operators on key parts inside each other.
- Recoded `MIN()` and `MAX()` to work properly with strings and `HAVING`.
- Changed default umask value for new files from `0664` to `0660`.
- Fixed problem with `LEFT JOIN` and constant expressions in the `ON` part.
- Added Italian error messages from `brenno@dewinter.com`.
- `configure` now works better on OSF/1 (tested on 4.0D).
- Added hooks to allow `LIKE` optimization with international character support.
- Upgraded `DBI` to 0.93.

D.6.11 Alterações na distribuição 3.21.23

- The following symbols are now reserved words: `TIME`, `DATE`, `TIMESTAMP`, `TEXT`, `BIT`, `ENUM`, `NO`, `ACTION`, `CHECK`, `YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, `SECOND`, `STATUS`, `VARIABLES`.
- Setting a `TIMESTAMP` to `NULL` in `LOAD DATA INFILE ...` didn't set the current time for the `TIMESTAMP`.
- Fix `BETWEEN` to recognise binary strings. Now `BETWEEN` is case-sensitive.
- Added `--skip-thread-priority` option to `mysqld`, for systems where `mysqld`'s thread scheduling doesn't work properly (BSDI 3.1).
- Added ODBC functions `DAYNAME()` and `MONTHNAME()`.
- Added function `TIME_FORMAT()`. This works like `DATE_FORMAT()`, but takes a time string (`'HH:MM:SS'`) as argument.
- Fixed unlikely(?) key optimiser bug when using `OR` operators of key parts inside `AND` expressions.
- Added `variables` command to `mysqladmin`.
- A lot of small changes to the binary releases.
- Fixed a bug in the new protocol from MySQL Version 3.21.20.
- Changed `ALTER TABLE` to work with Windows (Windows can't rename open files). Also fixed a couple of small bugs in the Windows version.
- All standard MySQL clients are now ported to MySQL for Windows.
- MySQL can now be started as a service on NT.

D.6.12 Alterações na distribuição 3.21.22

- Starting with this version, all MySQL distributions will be configured, compiled and tested with `crash-me` and the benchmarks on the following platforms: SunOS 5.6 sun4u, SunOS 5.5.1 sun4u, SunOS 4.14 sun4c, SunOS 5.6 i86pc, Irix 6.3 mips5k, HP-UX 10.20 hppa, AIX 4.2.1 ppc, OSF/1 V4.0 alpha, FreeBSD 2.2.2 i86pc and BSDI 3.1 i386.
- Fix `COUNT(*)` problems when the `WHERE` clause didn't match any records. (Bug from 3.21.17.)
- Removed that `NULL = NULL` is true. Now you must use `IS NULL` or `IS NOT NULL` to test whether a value is `NULL`. (This is according to SQL-99 but may break old applications that are ported from `mSQL`.) You can get the old behaviour by compiling with `-DmSQL_COMPLIANT`.
- Fixed bug that core dumped when using many `LEFT OUTER JOIN` clauses.
- Fixed bug in `ORDER BY` on string formula with possible `NULL` values.
- Fixed problem in range optimiser when using `<=` on sub index.
- Added functions `DAYOFYEAR()`, `DAYOFMONTH()`, `MONTH()`, `YEAR()`, `WEEK()`, `QUARTER()`, `HOUR()`, `MINUTE()`, `SECOND()` and `FIND_IN_SET()`.
- Added `SHOW VARIABLES` command.
- Added support of "long constant strings" from SQL-99:

```
mysql> SELECT 'first ' 'second';       -> 'first second'
```

- Upgraded Msql-Mysql-modules to 1.1825.
- Upgraded `mysqlaccess` to 2.02.
- Fixed problem with Russian character set and `LIKE`.
- Ported to OpenBSD 2.1.
- New Dutch error messages.

D.6.13 Alterações na distribuição 3.21.21a

- Configure changes for some operating systems.

D.6.14 Alterações na distribuição 3.21.21

- Fixed optimiser bug when using `WHERE data_field = date_field2 AND date_field2 = constant`.
- Added `SHOW STATUS` command.
- Removed '`manual.ps`' from the source distribution to make it smaller.

D.6.15 Alterações na distribuição 3.21.20

- Changed the maximum table name and column name lengths from 32 to 64.
- Aliases can now be of "any" length.
- Fixed `mysqladmin stat` to return the right number of queries.

- Changed protocol (downward compatible) to mark if a column has the `AUTO_INCREMENT` attribute or is a `TIMESTAMP`. This is needed for the new Java driver.
- Added Hebrew sorting order by Zeev Suraski.
- Solaris 2.6: Fixed `configure` bugs and increased maximum table size from 2G to 4G.

D.6.16 Alterações na distribuição 3.21.19

- Upgraded `DBD` to 1.1823. This version implements `mysql_use_result` in `DBD-Mysql`.
- Benchmarks updated for empress (by Luuk).
- Fixed a case of slow range searching.
- Configure fixes ('`Docs`' directory).
- Added function `REVERSE()` (by Zeev Suraski).

D.6.17 Alterações na distribuição 3.21.18

- Issue error message if client C functions are called in wrong order.
- Added automatic reconnect to the '`libmysql.c`' library. If a write command fails, an automatic reconnect is done.
- Small sort sets no longer use temporary files.
- Upgraded `DBI` to 0.91.
- Fixed a couple of problems with `LEFT OUTER JOIN`.
- Added `CROSS JOIN` syntax. `CROSS` is now a reserved word.
- Recoded `yacc/bison` stack allocation to be even safer and to allow MySQL to handle even bigger expressions.
- Fixed a couple of problems with the update log.
- `ORDER BY` was slow when used with key ranges.

D.6.18 Alterações na distribuição 3.21.17

- Changed documentation string of `--with-unix-socket-path` to avoid confusion.
- Added ODBC and SQL-99 style `LEFT OUTER JOIN`.
- The following are new reserved words: `LEFT`, `NATURAL`, `USING`.
- The client library now uses the value of the environment variable `MYSQL_HOST` as the default host if it's defined.
- `SELECT col_name, SUM(expr)` now returns `NULL` for `col_name` when there are matching rows.
- Fixed problem with comparing binary strings and `BLOB` values with ASCII characters over 127.
- Fixed lock problem: when freeing a read lock on a table with multiple read locks, a thread waiting for a write lock would have been given the lock. This shouldn't affect data integrity, but could possibly make `mysqld` restart if one thread was reading data that another thread modified.

- `LIMIT offset,count` didn't work in `INSERT ... SELECT`.
- Optimized key block caching. This will be quicker than the old algorithm when using bigger key caches.

D.6.19 Alterações na distribuição 3.21.16

- Added ODBC 2.0 & 3.0 functions `POWER()`, `SPACE()`, `COT()`, `DEGREES()`, `RADIANS()`, `ROUND(2 arg)` and `TRUNCATE()`.
- **Warning: Incompatible change!** `LOCATE()` parameters were swapped according to ODBC standard. Fixed.
- Added function `TIME_TO_SEC()`.
- In some cases, default values were not used for `NOT NULL` fields.
- Timestamp wasn't always updated properly in `UPDATE SET ...` statements.
- Allow empty strings as default values for `BLOB` and `TEXT`, to be compatible with `mysqldump`.

D.6.20 Alterações na distribuição 3.21.15

- **Warning: Incompatible change!** `mysqlperl` is now from Msql-Mysql-modules. This means that `connect()` now takes `host`, `database`, `user`, `password` arguments! The old version took `host`, `database`, `password`, `user`.
- Allow `DATE '1997-01-01'`, `TIME '12:10:10'` and `TIMESTAMP '1997-01-01 12:10:10'` formats required by SQL-99. **Warning: Incompatible change!** This has the unfortunate side-effect that you no longer can have columns named `DATE`, `TIME` or `TIMESTAMP`. :(Old columns can still be accessed through `tablename.columnname`!)
- Changed Makefiles to hopefully work better with BSD systems. Also, '`manual.dvi`' is now included in the distribution to avoid having stupid `make` programs trying to rebuild it.
- `readline` library upgraded to version 2.1.
- A new sortorder german-1. That is a normal ISO-Latin1 with a german sort order.
- Perl `DBI/DBD` is now included in the distribution. `DBI` is now the recommended way to connect to MySQL from Perl.
- New portable benchmark suite with `DBD`, with test results from `mSQL` 2.0.3, MySQL, PostgreSQL 6.2.1 and Solid server 2.2.
- `crash-me` is now included with the benchmarks; this is a Perl program designed to find as many limits as possible in an SQL server. Tested with `mSQL`, PostgreSQL, Solid and MySQL.
- Fixed bug in range-optimiser that crashed MySQL on some queries.
- Table and column name completion for `mysql` command-line tool, by Zeev Suraski and Andi Gutmans.
- Added new command `REPLACE` that works like `INSERT` but replaces conflicting records with the new record. `REPLACE INTO TABLE ... SELECT ...` works also.
- Added new commands `CREATE DATABASE db_name` and `DROP DATABASE db_name`.

- Added `RENAME` option to `ALTER TABLE`: `ALTER TABLE name RENAME TO new_name`.
- `make_binary_distribution` now includes '`libgcc.a`' in '`libmysqlclient.a`'. This should make linking work for people who don't have `gcc`.
- Changed `net_write()` to `my_net_write()` because of a name conflict with Sybase.
- New function `DAYOFWEEK()` compatible with ODBC.
- Stack checking and `bison` memory overrun checking to make MySQL safer with weird queries.

D.6.21 Alterações na distribuição 3.21.14b

- Fixed a couple of small `configure` problems on some platforms.

D.6.22 Alterações na distribuição 3.21.14a

- Ported to SCO Openserver 5.0.4 with FSU Pthreads.
- HP-UX 10.20 should work.
- Added new function `DATE_FORMAT()`.
- Added `NOT IN`.
- Added automatic removal of 'ODBC function conversions': `{fn now() }`
- Handle ODBC 2.50.3 option flags.
- Fixed comparison of `DATE` and `TIME` values with `NULL`.
- Changed language name from germany to german to be consistent with the other language names.
- Fixed sorting problem on functions returning a `FLOAT`. Previously, the values were converted to `INT` values before sorting.
- Fixed slow sorting when sorting on key field when using `key_column=constant`.
- Sorting on calculated `DOUBLE` values sorted on integer results instead.
- `mysql` no longer requires a database argument.
- Changed the place where `HAVING` should be. According to the SQL standards, it should be after `GROUP BY` but before `ORDER BY`. MySQL Version 3.20 incorrectly had it last.
- Added Sybase command `USE database` to start using another database.
- Added automatic adjusting of number of connections and table cache size if the maximum number of files that can be opened is less than needed. This should fix that `mysqld` doesn't crash even if you haven't done a `ulimit -n 256` before starting `mysqld`.
- Added lots of limit checks to make it safer when running with too little memory or when doing weird queries.

D.6.23 Alterações na distribuição 3.21.13

- Added retry of interrupted reads and clearing of `errno`. This makes Linux systems much safer!
- Fixed locking bug when using many aliases on the same table in the same `SELECT`.

- Fixed bug with `LIKE` on number key.
- New error message so you can check whether the connection was lost while the command was running or whether the connection was down from the start.
- Added `--table` option to `mysql` to print in table format. Moved time and row information after query result. Added automatic reconnect of lost connections.
- Added `!=` as a synonym for `<>`.
- Added function `VERSION()` to make easier logs.
- New multi-user test '`tests/fork_test.pl`' to put some strain on the thread library.

D.6.24 Alterações na distribuição 3.21.12

- Fixed `ftruncate()` call in MIT-pthreads. This made `isamchk` destroy the '`.ISM`' files on (Free)BSD 2.x systems.
- Fixed broken `__P_` patch in MIT-pthreads.
- Many memory overrun checks. All string functions now return `NULL` if the returned string should be longer than `max_allowed_packet` bytes.
- Changed the name of the `INTERVAL` type to `ENUM`, because `INTERVAL` is used in SQL-99.
- In some cases, doing a `JOIN + GROUP + INTO OUTFILE`, the result wasn't grouped.
- `LIKE` with `'_'` as last character didn't work. Fixed.
- Added extended SQL-99 `TRIM()` function.
- Added `CURTIME()`.
- Added `ENCRYPT()` function by Zeev Suraski.
- Fixed better `FOREIGN KEY` syntax skipping. New reserved words: `MATCH`, `FULL`, `PARTIAL`.
- `mysqld` now allows IP number and hostname for the `--bind-address` option.
- Added `SET CHARACTER SET cp1251_koi8` to enable conversions of data to and from the `cp1251_koi8` character set.
- Lots of changes for Windows 95 port. In theory, this version should now be easily portable to Windows 95.
- Changed the `CREATE COLUMN` syntax of `NOT NULL` columns to be after the `DEFAULT` value, as specified in the SQL-99 standard. This will make `mysqldump` with `NOT NULL` and default values incompatible with MySQL Version 3.20.
- Added many function name aliases so the functions can be used with ODBC or SQL-92 syntax.
- Fixed syntax of `ALTER TABLE tbl_name ALTER COLUMN col_name SET DEFAULT NULL`.
- Added `CHAR` and `BIT` as synonyms for `CHAR(1)`.
- Fixed core dump when updating as a user who has only `SELECT` privilege.
- `INSERT ... SELECT ... GROUP BY` didn't work in some cases. An `Invalid use of group function` error occurred.
- When using `LIMIT`, `SELECT` now always uses keys instead of record scan. This will give better performance on `SELECT` and a `WHERE` that matches many rows.
- Added Russian error messages.

D.6.25 Alterações na distribuição 3.21.11

- Configure changes.
- MySQL now works with the new thread library on BSD/OS 3.0.
- Added new group functions `BIT_OR()` and `BIT_AND()`.
- Added compatibility functions `CHECK` and `REFERENCES`. `CHECK` is now a reserved word.
- Added `ALL` option to `GRANT` for better compatibility. (`GRANT` is still a dummy function.)
- Added partly-translated Dutch error messages.
- Fixed bug in `ORDER BY` and `GROUP BY` with `NULL` columns.
- Added function `LAST_INSERT_ID()` SQL function to retrieve last `AUTO_INCREMENT` value. This is intended for clients to ODBC that can't use the `mysql_insert_id()` API function, but can be used by any client.
- Added `--flush-logs` option to `mysqladmin`.
- Added command `STATUS` to `mysql`.
- Fixed problem with `ORDER BY/GROUP BY` because of bug in `gcc`.
- Fixed problem with `INSERT ... SELECT ... GROUP BY`.

D.6.26 Alterações na distribuição 3.21.10

- New program `mysqlaccess`.
- `CREATE` now supports all ODBC types and the `mSQL` `TEXT` type. All ODBC 2.5 functions are also supported (added `REPEAT`). This provides better portability.
- Added text types `TINYTEXT`, `TEXT`, `MEDIUMTEXT` and `LONGTEXT`. These are actually `BLOB`types, but all searching is done in case-insensitive fashion.
- All old `BLOB` fields are now `TEXT` fields. This only changes that all searching on strings is done in case-sensitive fashion. You must do an `ALTER TABLE` and change the datatype to `BLOB` if you want to have tests done in case-sensitive fashion.
- Fixed some `configure` issues.
- Made the locking code a bit safer. Fixed very unlikely deadlock situation.
- Fixed a couple of bugs in the range optimiser. Now the new range benchmark `test-select` works.

D.6.27 Alterações na distribuição 3.21.9

- Added `--enable-unix-socket=pathname` option to `configure`.
- Fixed a couple of portability problems with include files.
- Fixed bug in range calculation that could return empty set when searching on multiple key with only one entry (very rare).
- Most things ported to FSU Pthreads, which should allow MySQL to run on SCO. Veja Seção 2.6.6.9 [SCO], Página 161.

D.6.28 Alterações na distribuição 3.21.8

- Works now in Solaris 2.6.
- Added handling of calculation of `SUM()` functions. For example, you can now use `SUM(column)/COUNT(column)`.
- Added handling of trigometric functions: `PI()`, `ACOS()`, `ASIN()`, `ATAN()`, `COS()`, `SIN()` and `TAN()`.
- New languages: Norwegian, Norwegian-ny and Portuguese.
- Fixed parameter bug in `net_print()` in '`procedure.cc`'.
- Fixed a couple of memory leaks.
- Now allow also the old `SELECT ... INTO OUTFILE` syntax.
- Fixed bug with `GROUP BY` and `SELECT` on key with many values.
- `mysql_fetch_lengths()` sometimes returned incorrect lengths when you used `mysql_use_result()`. This affected at least some cases of `mysqldump --quick`.
- Fixed bug in optimization of `WHERE const op field`.
- Fixed problem when sorting on `NULL` fields.
- Fixed a couple of 64-bit (Alpha) problems.
- Added `--pid-file=#` option to `mysqld`.
- Added date formatting to `FROM_UNIXTIME()`, originally by Zeev Suraski.
- Fixed bug in `BETWEEN` in range optimiser (did only test = of the first argument).
- Added machine-dependent files for MIT-pthreads i386-SCO. There is probably more to do to get this to work on SCO 3.5.

D.6.29 Alterações na distribuição 3.21.7

- Changed '`Makefile.am`' to take advantage of Automake 1.2.
- Added the beginnings of a benchmark suite.
- Added more secure password handling.
- Added new client function `mysql_errno()`, to get the error number of the error message. This makes error checking in the client much easier. This makes the new server incompatible with the 3.20.x server when running without `--old-protocol`. The client code is backward-compatible. More information can be found in the '`README`' file!
- Fixed some problems when using very long, illegal names.

D.6.30 Alterações na distribuição 3.21.6

- Fixed more portability issues (incorrect `sigwait` and `sigset` defines).
- `configure` should now be able to detect the last argument to `accept()`.

D.6.31 Alterações na distribuição 3.21.5

- Should now work with FreeBSD 3.0 if used with '`FreeBSD-3.0-libc_r-1.0.diff`', which can be found at `http://www.mysql.com/downloads/os-freebsd.html`.

- Added new `-O tmp_table_size=#` option to `mysqld`.
- New function `FROM_UNIXTIME(timestamp)` which returns a date string in `'YYYY-MM-DD HH:MM:SS'` format.
- New function `SEC_TO_TIME(seconds)` which returns a string in `'HH:MM:SS'` format.
- New function `SUBSTRING_INDEX()`, originally by Zeev Suraski.

D.6.32 Alterações na distribuição 3.21.4

- Should now configure and compile on OSF/1 4.0 with the DEC compiler.
- Configuration and compilation on BSD/OS 3.0 works, but due to some bugs in BSD/OS 3.0, `mysqld` doesn't work on it yet.
- Configuration and compilation on FreeBSD 3.0 works, but I couldn't get `pthread_create` to work.

D.6.33 Alterações na distribuição 3.21.3

- Added reverse check lookup of hostnames to get better security.
- Fixed some possible buffer overflows if filenames that are too long are used.
- `mysqld` doesn't accept hostnames that start with digits followed by a `'.'`, because the hostname may look like an IP number.
- Added `--skip-networking` option to `mysqld`, to allow only socket connections. (This will not work with MIT-pthreads!)
- Added check of too long table names for alias.
- Added check if database name is okay.
- Added check if too long table names.
- Removed incorrect `free()` that killed the server on `CREATE DATABASE` or `DROP DATABASE`.
- Changed some `mysqld -O` options to better names.
- Added `-O join_cache_size=#` option to `mysqld`.
- Added `-O max_join_size=#` option to `mysqld`, to be able to set a limit how big queries (in this case big = slow) one should be able to handle without specifying `SET SQL_BIG_SELECTS=1`. A # = is about 10 examined records. The default is "unlimited".
- When comparing a `TIME`, `DATE`, `DATETIME` or `TIMESTAMP` column to a constant, the constant is converted to a time value before performing the comparison. This will make it easier to get ODBC (particularly Access97) to work with the above types. It should also make dates easier to use and the comparisons should be quicker than before.
- Applied patch from Jochen Wiedmann that allows `query()` in `mysqlperl` to take a query with `\0` in it.
- Storing a timestamp with a 2-digit year (`YYMMDD`) didn't work.
- Fix that timestamp wasn't automatically updated if set in an `UPDATE` clause.
- Now the automatic timestamp field is the FIRST timestamp field.
- `SELECT * INTO OUTFILE`, which didn't correctly if the outfile already existed.

- `mysql` now shows the thread ID when starting or doing a reconnect.
- Changed the default sort buffer size from 2M to 1M.

D.6.34 Alterações na distribuição 3.21.2

- The range optimiser is coded, but only 85% tested. It can be enabled with `--new`, but it crashes core a lot yet...
- More portable. Should compile on AIX and alpha-digital. At least the `isam` library should be relatively 64-bit clean.
- New `isamchk` which can detect and fix more problems.
- New options for `isamlog`.
- Using new version of Automake.
- Many small portability changes (from the AIX and alpha-digital port) Better checking of pthread(s) library.
- czech error messages by `snajdr@pvt.net`.
- Decreased size of some buffers to get fewer problems on systems with little memory. Also added more checks to handle "out of memory" problems.
- `mysqladmin`: you can now do `mysqladmin kill 5,6,7,8` to kill multiple threads.
- When the maximum connection limit is reached, one extra connection by a user with the **process_acl** privilege is granted.
- Added `-O backlog=#` option to `mysqld`.
- Increased maximum packet size from 512K to 1024K for client.
- Almost all of the function code is now tested in the internal test suite.
- `ALTER TABLE` now returns warnings from field conversions.
- Port changed to 3306 (got it reserved from ISI).
- Added a fix for Visual FoxBase so that any schema name from a table specification is automatically removed.
- New function `ASCII()`.
- Removed function `BETWEEN(a,b,c)`. Use the standard SQL syntax instead: `expr BETWEEN expr AND expr`.
- MySQL no longer has to use an extra temporary table when sorting on functions or `SUM()` functions.
- Fixed bug that you couldn't use `tbl_name.field_name` in `UPDATE`.
- Fixed `SELECT DISTINCT` when using 'hidden group'. For example:

```
mysql> SELECT DISTINCT MOD(some_field,10) FROM test
    ->        GROUP BY some_field;
```

  Note: `some_field` is normally in the `SELECT` part. Standard SQL should require it.

D.6.35 Alterações na distribuição 3.21.0

- New reserved words used: `INTERVAL`, `EXPLAIN`, `READ`, `WRITE`, `BINARY`.
- Added ODBC function `CHAR(num,...)`.

- New operator `IN`. This uses a binary search to find a match.
- New command `LOCK TABLES tbl_name [AS alias] {READ|WRITE} ...`
- Added `--log-update` option to `mysqld`, to get a log suitable for incremental updates.
- New command `EXPLAIN SELECT ...` to get information about how the optimiser will do the join.
- For easier client code, the client should no longer use `FIELD_TYPE_TINY_BLOB`, `FIELD_TYPE_MEDIUM_BLOB`, `FIELD_TYPE_LONG_BLOB` or `FIELD_TYPE_VAR_STRING` (as previously returned by `mysql_list_fields`). You should instead only use `FIELD_TYPE_BLOB` or `FIELD_TYPE_STRING`. If you want exact types, you should use the command `SHOW FIELDS`.
- Added varbinary syntax: `0x######` which can be used as a string (default) or a number.
- `FIELD_TYPE_CHAR` is renamed to `FIELD_TYPE_TINY`.
- Changed all fields to C++ classes.
- Removed FORM struct.
- Fields with `DEFAULT` values no longer need to be `NOT NULL`.
- New field types:

  `ENUM` A string which can take only a couple of defined values. The value is stored as a 1-3 byte number that is mapped automatically to a string. This is sorted according to string positions!

  `SET` A string which may have one or many string values separated with ','. The string is stored as a 1-, 2-, 3-, 4- or 8-byte number where each bit stands for a specific set member. This is sorted according to the unsigned value of the stored packed number.
- Now all function calculation is done with `double` or `long long`. This will provide the full 64-bit range with bit functions and fix some conversions that previously could result in precision losses. One should avoid using `unsigned long long` columns with full 64-bit range (numbers bigger than 9223372036854775807) because calculations are done with `signed long long`.
- `ORDER BY` will now put `NULL` field values first. `GROUP BY` will also work with `NULL` values.
- Full `WHERE` with expressions.
- New range optimiser that can resolve ranges when some keypart prefix is constant. Example:

```
mysql> SELECT * FROM tbl_name
    ->          WHERE key_part_1="customer"
    ->          AND key_part_2>=10 AND key_part_2<=10;
```

D.7 Alterações na distribuição 3.20.x

Version 3.20 is quite old now, and should be avoided if possible. This information is kept here for historical purposes only.

Changes from 3.20.18 to 3.20.32b are not documented here because the 3.21 release branched here. And the relevant changes are also documented as changes to the 3.21 version.

D.7.1 Alterações na distribuição 3.20.18

- Added `-p#` (remove `#` directories from path) to `isamlog`. All files are written with a relative path from the database directory Now `mysqld` shouldn't crash on shutdown when using the `--log-isam` option.
- New `mysqlperl` version. It is now compatible with `msqlperl-0.63`.
- New `DBD` module available.
- Added group function `STD()` (standard deviation).
- The `mysqld` server is now compiled by default without debugging information. This will make the daemon smaller and faster.
- Now one usually only has to specify the `--basedir` option to `mysqld`. All other paths are relative in a normal installation.
- `BLOB` columns sometimes contained garbage when used with a `SELECT` on more than one table and `ORDER BY`.
- Fixed that calculations that are not in `GROUP BY` work as expected (SQL-99 extension). Example:

```
mysql> SELECT id,id+1 FROM table GROUP BY id;
```

- The test of using `MYSQL_PWD` was reversed. Now `MYSQL_PWD` is enabled as default in the default release.
- Fixed conversion bug which caused `mysqld` to core dump with Arithmetic error on SPARC-386.
- Added `--unbuffered` option to `mysql`, for new `mysqlaccess`.
- When using overlapping (unnecessary) keys and join over many tables, the optimiser could get confused and return 0 records.

D.7.2 Alterações na distribuição 3.20.17

- You can now use `BLOB` columns and the functions `IS NULL` and `IS NOT NULL` in the `WHERE` clause.
- All communication packets and row buffers are now allocated dynamically on demand. The default value of `max_allowed_packet` is now 64K for the server and 512K for the client. This is mainly used to catch incorrect packets that could trash all memory. The server limit may be changed when it is started.
- Changed stack usage to use less memory.
- Changed `safe_mysqld` to check for running daemon.
- The `ELT()` function is renamed to `FIELD()`. The new `ELT()` function returns a value based on an index: `FIELD()` is the inverse of `ELT()` Example: `ELT(2,"A","B","C")` returns `"B"`. `FIELD("B","A","B","C")` returns 2.
- `COUNT(field)`, where `field` could have a `NULL` value, now works.
- A couple of bugs fixed in `SELECT ... GROUP BY`.
- Fixed memory overrun bug in `WHERE` with many unoptimisable brace levels.
- Fixed some small bugs in the grant code.

- If hostname isn't found by `get_hostname`, only the IP is checked. Previously, you got `Access denied`.
- Inserts of timestamps with values didn't always work.
- `INSERT INTO ... SELECT ... WHERE` could give the error `Duplicated field`.
- Added some tests to `safe_mysqld` to make it "safer".
- `LIKE` was case-sensitive in some places and case-insensitive in others. Now `LIKE` is always case-insensitive.
- '`mysql.cc`': Allow `'#'` anywhere on the line.
- New command `SET SQL_SELECT_LIMIT=#`. See the FAQ for more details.
- New version of the `mysqlaccess` script.
- Change `FROM_DAYS()` and `WEEKDAY()` to also take a full `TIMESTAMP` or `DATETIME` as argument. Before they only took a number of type `YYYYMMDD` or `YYMMDD`.
- Added new function `UNIX_TIMESTAMP(timestamp_column)`.

D.7.3 Alterações na distribuição 3.20.16

- More changes in MIT-pthreads to get them safer. Fixed also some link bugs at least in SunOS.
- Changed `mysqld` to work around a bug in MIT-pthreads. This makes multiple small `SELECT` operations 20 times faster. Now `lock_test.pl` should work.
- Added `mysql_FetchHash(handle)` to `mysqlperl`.
- The `mysqlbug` script is now distributed built to allow for reporting bugs that appear during the build with it.
- Changed '`libmysql.c`' to prefer `getpwuid()` instead of `cuserid()`.
- Fixed bug in `SELECT` optimiser when using many tables with the same column used as key to different tables.
- Added new `latin2` and Russian `KOI8` character tables.
- Added support for a dummy `GRANT` command to satisfy Powerbuilder.

D.7.4 Alterações na distribuição 3.20.15

- Fixed fatal bug `packets out of order` when using MIT-pthreads.
- Removed possible loop when a thread waits for command from client and `fcntl()` fails. Thanks to Mike Bretz for finding this bug.
- Changed alarm loop in '`mysqld.cc`' because shutdown didn't always succeed in Linux.
- Removed use of `termbits` from '`mysql.cc`'. This conflicted with `glibc` 2.0.
- Fixed some syntax errors for at least BSD and Linux.
- Fixed bug when doing a `SELECT` as superuser without a database.
- Fixed bug when doing `SELECT` with group calculation to outfile.

D.7.5 Alterações na distribuição 3.20.14

- If one gives `-p` or `--password` option to `mysql` without an argument, the user is solicited for the password from the tty.
- Added default password from `MYSQL_PWD` (by Elmar Haneke).
- Added command `kill` to `mysqladmin` to kill a specific MySQL thread.
- Sometimes when doing a reconnect on a down connection this succeeded first on second try.
- Fixed adding an `AUTO_INCREMENT` key with `ALTER_TABLE`.
- `AVG()` gave too small value on some `SELECT` statements with `GROUP BY` and `ORDER BY`.
- Added new `DATETIME` type (by Giovanni Maruzzelli `maruzz@matrice.it`).
- Fixed that defining `DONT_USE_DEFAULT_FIELDS` works.
- Changed to use a thread to handle alarms instead of signals on Solaris to avoid race conditions.
- Fixed default length of signed numbers. (George Harvey `georgeh@pinacl.co.uk`.)
- Allow anything for `CREATE INDEX`.
- Add prezeros when packing numbers to `DATE`, `TIME` and `TIMESTAMP`.
- Fixed a bug in `OR` of multiple tables (gave empty set).
- Added many patches to MIT-pthreads. This fixes at least one lookup bug.

D.7.6 Alterações na distribuição 3.20.13

- Added standard SQL-92 `DATE` and `TIME` types.
- Fixed bug in `SELECT` with `AND-OR` levels.
- Added support for Slovenian characters. The '`Contrib`' directory contains source and instructions for adding other character sets.
- Fixed bug with `LIMIT` and `ORDER BY`.
- Allow `ORDER BY` and `GROUP BY` on items that aren't in the `SELECT` list. (Thanks to Wim Bonis `bonis@kiss.de`, for pointing this out.)
- Allow setting of timestamp values in `INSERT`.
- Fixed bug with `SELECT ... WHERE ... = NULL`.
- Added changes for `glibc` 2.0. To get `glibc` to work, you should add the '`gibc-2.0-sigwait-patch`' before compiling `glibc`.
- Fixed bug in `ALTER TABLE` when changing a `NOT NULL` field to allow `NULL` values.
- Added some SQL-92 synonyms as field types to `CREATE TABLE`. `CREATE TABLE` now allows `FLOAT(4)` and `FLOAT(8)` to mean `FLOAT` and `DOUBLE`.
- New utility program `mysqlaccess` by `Yves.Carlier@rug.ac.be`. This program shows the access rights for a specific user and the grant rows that determine this grant.
- Added `WHERE const op field` (by `bonis@kiss.de`).

D.7.7 Alterações na distribuição 3.20.11

- When using `SELECT ... INTO OUTFILE`, all temporary tables are ISAM instead of HEAP to allow big dumps.
- Changed date functions to be string functions. This fixed some "funny" side effects when sorting on dates.
- Extended `ALTER TABLE` for SQL-92 compliance.
- Some minor compatibility changes.
- Added `--port` and `--socket` options to all utility programs and `mysqld`.
- Fixed MIT-pthreads `readdir_r()`. Now `mysqladmin create database` and `mysqladmin drop database` should work.
- Changed MIT-pthreads to use our `tempnam()`. This should fix the "sort aborted" bug.
- Added sync of records count in `sql_update`. This fixed slow updates on first connection. (Thanks to Vaclav Bittner for the test.)

D.7.8 Alterações na distribuição 3.20.10

- New insert type: `INSERT INTO ... SELECT ...`
- `MEDIUMBLOB` fixed.
- Fixed bug in `ALTER TABLE` and `BLOB` values.
- `SELECT ... INTO OUTFILE` now creates the file in the current database directory.
- `DROP TABLE` now can take a list of tables.
- Oracle synonym `DESCRIBE` (`DESC`).
- Changes to `make_binary_distribution`.
- Added some comments to installation instructions about `configure`'s C++ link test.
- Added `--without-perl` option to `configure`.
- Lots of small portability changes.

D.7.9 Alterações na distribuição 3.20.9

- `ALTER TABLE` didn't copy null bit. As a result, fields that were allowed to have `NULL` values were always `NULL`.
- `CREATE` didn't take numbers as `DEFAULT`.
- Some compatibility changes for SunOS.
- Removed '`config.cache`' from old distribution.

D.7.10 Alterações na distribuição 3.20.8

- Fixed bug with `ALTER TABLE` and multi-part keys.

D.7.11 Alterações na distribuição 3.20.7

- New commands: `ALTER TABLE`, `SELECT ... INTO OUTFILE` and `LOAD DATA INFILE`.
- New function: `NOW()`.
- Added new field `File_priv` to `mysql/user` table.
- New script `add_file_priv` which adds the new field `File_priv` to the `user` table. This script must be executed if you want to use the new `SELECT ... INTO` and `LOAD DATA INFILE ...` commands with a version of MySQL earlier than 3.20.7.
- Fixed bug in locking code, which made `lock_test.pl` test fail.
- New files '`NEW`' and '`BUGS`'.
- Changed '`select_test.c`' and '`insert_test.c`' to include '`config.h`'.
- Added `status` command to `mysqladmin` for short logging.
- Increased maximum number of keys to 16 and maximum number of key parts to 15.
- Use of sub keys. A key may now be a prefix of a string field.
- Added `-k` option to `mysqlshow`, to get key information for a table.
- Added long options to `mysqldump`.

D.7.12 Alterações na distribuição 3.20.6

- Portable to more systems because of MIT-pthreads, which will be used automatically if `configure` cannot find a `-lpthreads` library.
- Added GNU-style long options to almost all programs. Test with *`program`* `--help`.
- Some shared library support for Linux.
- The FAQ is now in '`.texi`' format and is available in '`.html`', '`.txt`' and '`.ps`' formats.
- Added new SQL function `RAND([init])`.
- Changed `sql_lex` to handle `\0` unquoted, but the client can't send the query through the C API, because it takes a str pointer. You must use `mysql_real_query()` to send the query.
- Added API function `mysql_get_client_info()`.
- `mysqld` now uses the `N_MAX_KEY_LENGTH` from '`nisam.h`' as the maximum allowable key length.
- The following now works:

  ```
  mysql> SELECT filter_nr,filter_nr FROM filter ORDER BY filter_nr;
  ```

  Previously, this resulted in the error: `Column: 'filter_nr' in order clause is ambiguous`.
- `mysql` now outputs `'\0'`, `'\t'`, `'\n'` and `'\\'` when encountering ASCII 0, tab, newline or `'\'` while writing tab-separated output. This is to allow printing of binary data in a portable format. To get the old behaviour, use `-r` (or `--raw`).
- Added german error messages (60 of 80 error messages translated).
- Added new API function `mysql_fetch_lengths(MYSQL_RES *)`, which returns an array of column lengths (of type `uint`).

- Fixed bug with `IS NULL` in `WHERE` clause.
- Changed the optimiser a little to get better results when searching on a key part.
- Added `SELECT` option `STRAIGHT_JOIN` to tell the optimiser that it should join tables in the given order.
- Added support for comments starting with `'--'` in '`mysql.cc`' (Postgres syntax).
- You can have `SELECT` expressions and table columns in a `SELECT` which are not used in the group part. This makes it efficient to implement lookups. The column that is used should be a constant for each group because the value is calculated only once for the first row that is found for a group.

```
mysql> SELECT id,lookup.text,SUM(*) FROM test,lookup
    ->        WHERE test.id=lookup.id GROUP BY id;
```

- Fixed bug in `SUM(function)` (could cause a core dump).
- Changed `AUTO_INCREMENT` placement in the SQL query:

```
INSERT INTO table (auto_field) VALUES (0);
```

  inserted 0, but it should insert an `AUTO_INCREMENT` value.
- '`mysqlshow.c`': Added number of records in table. Had to change the client code a little to fix this.
- `mysql` now allows doubled `''` or `""` within strings for embedded `'` or `"`.
- New math functions: `EXP()`, `LOG()`, `SQRT()`, `ROUND()`, `CEILING()`.

D.7.13 Alterações na distribuição 3.20.3

- The `configure` source now compiles a thread-free client library `-lmysqlclient`. This is the only library that needs to be linked with client applications. When using the binary releases, you must link with `-lmysql -lmysys -ldbug -lmystrings` as before.
- New `readline` library from `bash-2.0`.
- LOTS of small changes to `configure` and makefiles (and related source).
- It should now be possible to compile in another directory using `VPATH`. Tested with GNU Make 3.75.
- `safe_mysqld` and `mysql.server` changed to be more compatible between the source and the binary releases.
- `LIMIT` now takes one or two numeric arguments. If one argument is given, it indicates the maximum number of rows in a result. If two arguments are given, the first argument indicates the offset of the first row to return, the second is the maximum number of rows. With this it's easy to do a poor man's next page/previous page WWW application.
- Changed name of SQL function `FIELDS()` to `ELT()`. Changed SQL function `INTERVALL()` to `INTERVAL()`.
- Made `SHOW COLUMNS` a synonym for `SHOW FIELDS`. Added compatibility syntax `FRIEND KEY` to `CREATE TABLE`. In MySQL, this creates a non-unique key on the given columns.
- Added `CREATE INDEX` and `DROP INDEX` as compatibility functions. In MySQL, `CREATE INDEX` only checks if the index exists and issues an error if it doesn't exist. `DROP INDEX` always succeeds.

- '`mysqladmin.c`': added client version to version information.
- Fixed core dump bug in `sql_acl` (core on new connection).
- Removed `host`, `user` and `db` tables from database `test` in the distribution.
- `FIELD_TYPE_CHAR` can now be signed (-128 to 127) or unsigned (0 to 255) Previously, it was always unsigned.
- Bug fixes in `CONCAT()` and `WEEKDAY()`.
- Changed a lot of source to get `mysqld` to be compiled with SunPro compiler.
- SQL functions must now have a `'('` immediately after the function name (no intervening space). For example, `'USER('` is regarded as beginning a function call, and `'USER ('` is regarded as an identifier `USER` followed by a `'('`, not as a function call.

D.7.14 Alterações na distribuição 3.20.0

- The source distribution is done with `configure` and Automake. It will make porting much easier. The `readline` library is included in the distribution.
- Separate client compilation: the client code should be very easy to compile on systems which don't have threads.
- The old Perl interface code is automatically compiled and installed. Automatic compiling of `DBD` will follow when the new `DBD` code is ported.
- Dynamic language support: `mysqld` can now be started with Swedish or English (default) error messages.
- New functions: `INSERT()`, `RTRIM()`, `LTRIM()` and `FORMAT()`.
- `mysqldump` now works correctly for all field types (even `AUTO_INCREMENT`). The format for `SHOW FIELDS FROM tbl_name` is changed so the `Type` column contains information suitable for `CREATE TABLE`. In previous releases, some `CREATE TABLE` information had to be patched when re-creating tables.
- Some parser bugs from 3.19.5 (`BLOB` and `TIMESTAMP`) are corrected. `TIMESTAMP` now returns different date information depending on its create length.
- Changed parser to allow a database, table or field name to start with a number or `'_'`.
- All old C code from Unireg changed to C++ and cleaned up. This makes the daemon a little smaller and easier to understand.
- A lot of small bug fixes done.
- New '`INSTALL`' files (not final version) and some information regarding porting.

D.8 Alterações na distribuição 3.19.x

Version 3.19 is quite old now, and should be avoided if possible. This information is kept here for historical purposes only.

D.8.1 Alterações na distribuição 3.19.5

- Some new functions, some more optimization on joins.
- Should now compile clean on Linux (2.0.x).

- Added functions `DATABASE()`, `USER()`, `POW()`, `LOG10()` (needed for ODBC).
- In a `WHERE` with an `ORDER BY` on fields from only one table, the table is now preferred as first table in a multi-join.
- `HAVING` and `IS NULL` or `IS NOT NULL` now works.
- A group on one column and a sort on a group function (`SUM()`, `AVG()`...) didn't work together. Fixed.
- `mysqldump`: Didn't send password to server.

D.8.2 Alterações na distribuição 3.19.4

- Fixed horrible locking bug when inserting in one thread and reading in another thread.
- Fixed one-off decimal bug. 1.00 was output as 1.0.
- Added attribute `'Locked'` to process list as information if a query is locked by another query.
- Fixed full magic timestamp. Timestamp length may now be 14, 12, 10, 8, 6, 4 or 2 bytes.
- Sort on some numeric functions could sort incorrectly on last number.
- `IF(arg,syntax_error,syntax_error)` crashed.
- Added functions `CEILING()`, `ROUND()`, `EXP()`, `LOG()` and `SQRT()`.
- Enhanced `BETWEEN` to handle strings.

D.8.3 Alterações na distribuição 3.19.3

- Fixed `SELECT` with grouping on `BLOB` columns not to return incorrect `BLOB` info. Grouping, sorting and distinct on `BLOB` columns will not yet work as expected (probably it will group/sort by the first 7 characters in the `BLOB`). Grouping on formulas with a fixed string size (use `MID()` on a `BLOB`) should work.
- Ao se fazer um full join (sem chave diretas) em tabelas múltiplas com campos `BLOB`, o `BLOB` vinha como lixo na saída.
- Corrigido `DISTINCT` com colunas calculadas.

Apêndice E Portando para Outros Sistemas

Este apêndice lhe ajudará a portar o MySQL para outros sistemas operacionais. Primeiro verifique a lista de sistemas operacionais atualemente suportados. Veja Seção 2.2.3 [Qual SO], Página 78. Se você criou uma nova portabilidade do MySQL, por favor, deixe nos conhecê-la para que possamos lista-la aqui e em nosso site web. (`http://www.mysql.com/`), recomendando-a a outros usuários.

Nota: se voce criou uma nova portabilidade para o MySQL, você está livre para distribuí-la sob a licença `GPL`, mas isto não te dá os direitos autorais do MySQL.

Uma biblioteca thread Posix funcionando é necessária para o servidor. No Solaris 2.5 nós usamos Pthreads da Sun (o suporte da thread nativa na versão 2.4 e anterior não está boa o suficiente), no Linux usamos LinuxThreads criada por Xavier Leroy, `Xavier.Leroy@inria.fr`.

A parte difícil de portar para uma nova variante Unix sem um bom suporte a thread nativa é, provavelmente, portar par MIT-pthreads. Veja '`mit-pthreads/README`' e Programando em Thhredas POSIX (`http://www.humanfactor.com/pthreads/`).

Até o MySQL 4.0.2, a distribuição do MySQL incluiu uma versão "remendada" do Pthreads de Chris Provenzano do MIT (veja o site de MIT Pthreads em `http://www.mit.edu/afs/sipb/project/pthreads/` e uma introdução a programação em `http://www.mit.edu:8001/people/proven/IAP_2000/`). Eles podem ser usadas por alguns sistemas operacionais que não têm threads POSIX. Veja Seção 2.3.6 [MIT-pthreads], Página 106.

Também é possível usar outro pacote de threads no nível do usuário chamado FSU Pthreads (veja `http://moss.csc.ncsu.edu/~mueller/pthreads/`). Esta implementação está usada para portar para o SCO.

Veja os programas '`thr_lock.c`' e '`thr_alarm.c`' no diretório '`mysys`' para alguns testes/exemplos destes problemas.

Tanto o servidor quanto o cliente precisam de um compilador C++ funcionanado. Nós usamos `gcc` em muitas plataormas. Outros compiladores que sabemos que funciona são o SPARCworksm Sun Forte, Irix `cc`, HP-UX `aCC`, IBM AIX `xlC_r`), Intel `ecc` e Compaq `cxx`).

Para compilar apenas o cliente use `./configure --without-server`.

Atualmente não hé nenhum suporte para compilação só do servidor, nem está em pauta a sua adição a menos que alguém tenha uma boa razão para isto.

Se você quiser/precisar de alterar qualquer '`Makefile`' ou o script do configure você também precisará do GNU Automake e Autoconf. Veja Seção 2.3.4 [Instalando a árvore de fontes], Página 100.

Todos os passos necessários para refazer tudo desde os arquivos mais básicos.

```
/bin/rm */.deps/*.P
/bin/rm -f config.cache
aclocal
autoheader
aclocal
automake
autoconf
```

```
./configure --with-debug=full --prefix='your installation directory'

# O  makefile gerado acima precsa do GNU make 3.75 ou mais novo.
# (chamado gmake abaixo)
gmake clean all install init-db
```

Se você encontrar problemas com uma nova portabilidade, você ter que fazer alguma depuração do MySQL! Veja Seção E.1 [Depurando o servidor], Página 1079.

Nota: antes de iniciar a depuração do `mysqld`, obtenha primeiro os programas de teste `mysys/thr_alarm` e `mysys/thr_lock` para funcionar. Isto asegurará que sus instalação da thread tem pelo menos uma chance remota de funcionar.

E.1 Depurando um Servidor MySQL

Se você estiver usando uma funcionalidade que é muito nova no MySQL, você pode tentar executar o `mysqld` com `--skip-new` (que desabilitará todas novas funcionalidades com pontecialidade de ero) ou com `--safe-mode` que desabilita várias otimizações que podem criar problemas. Veja Seção A.4.1 [Falhas], Página 925.

Se o `mysqld` não quiser iniciar, você deve verificar se você não tem qualquer arquivo '`my.cnf`' que interfere com sua configuração. Você pode verificar seus argumento do '`my.cnf`' com `mysqld --print-defaults` e evitar usá-los iniciando com `mysqld --no-defaults ...`.

Se o `mysqld` começa a consumir CPU ou memória ou se ele ficar lento, você pode usar o `mysqladmin processlist status` para achar alguém que esteja executando uma consulta que demore algum tempo. POde ser uma boa idéia executar `mysqladmin -i10 processlist status` em alguma janela se você estiver tendo problemas de desempenho ou problemas com novos clientes que não podem conectar.

O comando `mysqladmin debug` irá trazer alguma informação sobre as em uso, memória usada e uso das consultas no arquivo de log do mysql. Isto pode ajudar a resolver problemas. Este comando também fornece informações úteis mesmo se você não tiver compilado MySQL para depuração!

Se o problema é que algumas tabelas estão ficando maior e mais lentas você deve tentar otimizar a tabela com `OPTIMIZE TABLE` ou `myisamchk`. Veja Cap"ptexi tulo 4 [Administração de Banco de Dados MySQL], Página 207. Você também deve tentar verificar as consultas lentas com `EXPLAIN`.

Você também deve ler a seção específica do SO neste manual para saber sobre problemas que podem ser únicos em seu ambiente. Veja Seção 2.6 [Notas Específicas do Sistema Operacional], Página 133.

E.1.1 Compilando o MYSQL para Depuração

Se você tiver um problema específico, você sempre pode tentar depurar o MySQL. Para fazer isto você deve configurar o MySQL com a opção `--with-debug` ou `--with-debug=full`. Você pode verificar se o MySQL foi compilado com depuração executando: `mysqld --help`. Se o parâmetro `--debug` estiver listado entre as opções então você têm a depuração habilitada. `mysqladmin ver` também lista a versão do `mysqld` como `mysql ... --debug` neste caso.

se você estiver usando gcc ou egcs, a configuração recomendada é:

```
CC=gcc CFLAGS="-O2" CXX=gcc CXXFLAGS="-O2 -felide-constructors \
   -fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
   --with-debug --with-extra-charsets=complex
```

Isto evitará problemas com a biblioteca `libstdc++` e com exceções C++ (muitos compiladores têm problemas com exceções C++ no código da thread) e compila uma versão MySQL com suporte para todos os conjuntos caracter.

Se você suspeita de um erro despejo de memória, você pode configurar o o MySQL com `--with-debug=full`, que irá instalar verificar de alocação de memória (`SAFEMALLOC`). No entanto, a execução com `SAFEMALLOC` é um pouco lenta, assim se você tiver problemas de desempenho você deve iniciar o `mysqld` com a opção `--skip-safemalloc`. Isto disabilitará a verificação de despejo de momória para cada chamada a `malloc()` e `free()`.

Se o `mysqld` parar de falhar quando você compilar com `--with-debug`, você provavelmente encontrou um erro de compilação ou erro de tempo dentro do MySQL. Neste caso você pode tentar adicionar `-g` às variáveis `CFLAGS` e `CXXFLAGS` acima e não usar `--with-debug`. Se agora o `mysqld` morre, você pode pelo menos executá-lo com `gdb` ou usar o `gdb` no arquivo core para descobrir que aconteceu.

Quando você configura o MySQL para depuração você habilita automaticamente diversas funções de vericação de segurança extra que monitora a saúde do `mysqld`. Se eles encontrarem algo "inesperado", uma entrada será encrita no `stderr`, que `mysqld_safe` direciona para o log de erros! Isto também significa que se você estiver tendo alguns problemas inexperados com o MySQL e estiver usando uma distribuição fonte, a primeira coisa que você deve fazer é configurar o MySQL para depuração! (A segunda coisa é enviar uma mensagem para a lista de email do MySQL e pedir ajuda. Veja Seção 1.7.1.1 [Mailing-list], Página 33. Por favor, use o script `mysqlbug` para todos os relatos de bug e questões referentes a versão do MySQL que você está usando!

Na distribuição do MySQL para Windows, `mysqld.exe` é, por padrão, compilado com suporte a arquivos trace.

E.1.2 Criando Arquivos Trace (Rastreamento)

Se o servidor `mysqld` não inicia ou se você pode fazer o servidor `mysqld` falhar rapidamente, você pode tentar criar um arquivo trace para encontrar o problema.

Para fazer isto você tem que ter um `mysqld` compilado para depuração. Você pode verificar isto executando `mysqld -V`. Se o número da versão finaliza com `-debug`, ele está compilado com suporte a arquivos trace.

Inicie o servidor `mysqld` com um log trace em '`/tmp/mysqld.trace`' (ou '`C:\mysqld.trace`' no Windows):

```
mysqld --debug
```

No Windows você também deve usar o parâmetro `--standalone` para não iniciar o `mysqld` como um serviço:

Em uma janela de console faça:

```
mysqld --debug --standalone
```

Depois disso você pode usar a ferramenta de linha de comando `mysql.exe` em uma segunda janela de console para reproduzir o problema. Você pode finalizar o servidor `mysqld` acima com `mysqladmin shutdown`.

Note que o arquivo trace será **muito grande**! Se você quiser ter um arquivo trace menor, você pode usar algo como:

```
mysqld --debug=d,info,error,query,general,where:0,/tmp/mysqld.trace
```

que apenas exibe informações com a maioria das tags interrassants em '`/tmp/mysqld.trace`'.

Se você fizer um relatório de bug sobre isto, por favor só envie as linhas do trace para a lista de email apropriada quando algo parecee estar errado! Se você não puder localizar o local errado, você pode fazer um ftp do arquivo trace, junto com um relatório de bug completo, para `ftp://support.mysql.com/pub/mysql/secret/` para que assim um desenvolvedor do MySQL possa dar uma olhada nele.

O arquivo trace é feito com o pacote **DBUG** de Fred Fish. Veja Seção E.3 [O pacote DBUG], Página 1085.

E.1.3 Depurando o mysqld no gdb

Na maioria dos sistemas você também pode iniciar o `mysqld` a partir do `gdb` para obter mais informações se o `mysqld` falhar.

Com uma versão antiga do `gdb` no Linux você deve usar `run --one-thread` se você quiser estar apto a depurar a thread `mysqld`. Neste caso você só pode ter uma thread ativa por vez. Nós recomendamos que você atualize para gdb 5.1 ASAP já que a depuração da thread funciona muito melhor com esta versão!

Ao executar o `mysqld` com gdb, você deve disabilitar a pilha de rastreamento com `--skip-stack-trace` para estar apto a conseguir segmentation fault com gdb.

É muito difícil depurar o MySQL no `gdb` se você fizer muitas conexões o tempo todo já que `gdb` não libera a memória para threads antigas. Você pode evitar este problema iniciando `mysqld` com `-O thread_cache_size= 'max_connections +1'`. Na maioria dos casos só o uso de `-O thread_cache_size=5'` já ajuda muito!

Se você quiser um tiver um core dump no Linux quando o `mysqld` morre com um sinal SIGSEGV, você pode iniciar o `mysqld` com a opção `--core-file`. Este arquivo core pode ser usado para fazer um rastreamento que pode lhe ajudar a descobrir porque o `mysqld` morreu:

```
shell> gdb mysqld core
gdb>   backtrace full
gdb>   exit
```

Veja Seção A.4.1 [Falhas], Página 925.

Se você estiver usando gdb 4.17.x ou acima no Linux, você deve instalar um arquivo '`.gdb`', com a seguinte informação, em seu diretório atual:

```
set print sevenbit off
handle SIGUSR1 nostop noprint
handle SIGUSR2 nostop noprint
handle SIGWAITING nostop noprint
```

```
handle SIGLWP nostop noprint
handle SIGPIPE nostop
handle SIGALRM nostop
handle SIGHUP nostop
handle SIGTERM nostop noprint
```

Se você tiver problemas depurando threads com gdb, você deve fazer o download do gdb 5.x e experimentá-lo. A nova versão do gdb tem um tratamento de threads bem melhorado.

Aqui está um exemplo de como depurar o mysqld:

```
shell> gdb /usr/local/libexec/mysqld
gdb> run
...
backtrace full # Faça isto quando o mysqld falhar
```

Inclua a saída acima e uma email gerado com `mysqlbug` e envie-o para lista de email do MySQL. Veja Seção 1.7.1.1 [Mailing-list], Página 33.

Se o `mysqld` travar você pode usar algumas ferramentas de sistema como `strace` ou `/usr/proc/bin/pstack` para exeminar onde `mysqld` travou.

```
strace /tmp/log libexec/mysqld
```

Se você estiver usando a interface Perl `DBI`, você pode habilitar a informação de depuação usando o método `trace` ou definindo a variável de ambiente `DBI_TRACE`. Veja Seção 12.5.2 [Classe Perl `DBI`], Página 881.

E.1.4 Usando Stack Trace

Em alguns sistemas operacionais, o log de erro irá conter um stack trace se `mysqld` finalizar inesperadmente. Você pode usá-lo para descobrir onde (e talvez por que) o `mysqld` finalizou. Veja Seção 4.10.1 [Log de erro], Página 372. Para obter um stack trace, você não deve compilar o `mysqld` com a opção `-fomit-frame-pointer` para gcc. Veja Seção E.1.1 [Compilando para depuração], Página 1079.

Se o arquivo de erro conter algo como o seguinte:

```
mysqld got signal 11;
The manual section 'Debugging a MySQL server' tells you how to use a
stack trace and/or the core file to produce a readable backtrace that may
help in finding out why mysqld died
Attemping backtrace. You can use the following information to find out
where mysqld died.  If you see no messages after this, something went
terribly wrong
stack range sanity check, ok, backtrace follows
0x40077552
0x81281a0
0x8128f47
0x8127be0
0x8127995
0x8104947
0x80ff28f
0x810131b
```

```
0x80ee4bc
0x80c3c91
0x80c6b43
0x80c1fd9
0x80c1686
```

você pode descobrir onde o `mysqld` finalizou fazendo o seguinte:

1. Copie os números acima em um arquivo, por exemplo '`mysqld.stack`'.
2. Faça um arquivo de símbolos para o servidor `mysqld`:

   ```
   nm -n libexec/mysqld > /tmp/mysqld.sym
   ```

   Note que a maioria das distribuições binárias do MySQL (exceto para o pacotes de "depuração" onde as informações são incluídas dentro dos binários) já possuem o arquivo acima, chamado `mysqld.sym.gz`. Neste caso você pode simplesmente desempacotá-lo fazendo:

   ```
   gunzip < bin/mysqld.sym.gz > /tmp/mysqld.sym
   ```

3. Execute `resolve_stack_dump -s /tmp/mysqld.sym -n mysqld.stack`.

   Isto exibirá a onde o `mysqld` finalizou. Se isto não lhe ajuda a descobrir o porque o `mysqld` morreu, você deve fazer um relato de erro e incluir a saída do comando acima no relatório.

   Note no entanto que na maioria dos casos, termos apenas um stack trace, não nos ajudará a encontrar a razão do problema. Para estarmos apto a localizar o erro ou fornecer um modo de contorná-lo, precisariamos, na maioria dos casos, conhecer a consulta que matou o `mysqld` e de preferência um caso de teste para que possamos repetir o problema! Veja Seção 1.7.1.3 [Relato de erros], Página 36.

E.1.5 Usando Arquivos de Log para Encontrar a Causa dos Erros no mysqld

Note que antes de iniciarmos o `mysqld` com `--log` você deve verificar todas as suas tabelas com `myisamchk`. Veja Cap“ptexi tulo 4 [Administração do Banco de Dados MySQL], Página 207.

Se o `mysqld` morre ou trava, você deve iniciar o `mysqld` com `--log`. Quando o `mysqld` morre de novo, você pode examinar o fim do arquivo de log para a consulta que matou o `mysqld`.

Se você estiver usando `--log` sem um nome de arquivo, o log é armazenado no diretório do banco de dados como 'nomemaquina'.log. Na maioria dos casos é a última consulta no arquivo de log que matou `mysqld`, mas se possível você deve verificar isto reiniciando o `mysqld` e executando a consulta encontrada por meio da ferramenta de linha de comando `mysql`. Se isto funcionar, você também deve testar todas as consultas complicadas que não completaram.

Você também pode tentar o comando `EXPLAIN` em todas as instruções `SELECT` que levam muito tempo para assegurar que o `mysqld` está usando índices apropriados. Veja Seção 5.2.1 [`EXPLAIN`], Página 424.

Você pode encontrar as consultas que levam muito twempo para executar iniciando o `mysqld` com `--log-slow-queries`. Veja Seção 4.10.5 [Log de consultas lentas], Página 377.

Se você encontrar o texto `mysqld restarted` no arquivo de registro de erro (normalmente chamado '`hostname.err`') você provavelmente encontrou uma consulta que provocou a falha no `mysqld`. Se isto acontecer você deve verificar todas as suas tabelas com `myisamchk` (veja Cap"ptexi tulo 4 [Administração do Banco de Dados MySQL], Página 207) e testar a consulta no arquivo de log do MySQL para ver se ela não funcionou. Se você encontrar tal consulta, tente primeiro atualizar para uma versão mais nova do MySQL. Se isto não ajudar e você não puder encontrar nada no arquivo de mensagem `mysql`, você deve relatar o erro para uma lista de email do MySQL. As listas de email estão descritas em `http://lists.mysql.com/`, que também possui os links para as listas de arquivos online.

Se você iniciou o `mysqld` com `myisam-recover`, o MySQL verificará automaticamente e tentará reparar as tabelas `MyISAM` se elas estiverem marcadas como 'não fechadas apropriadamente' ou 'com falha'. Se isto acontecer, o MySQL irá escrever uma entrada '`Warning: Checking table ...`' no arquivo '`nomemaquina.err`', a qual é seguida por `Warning: Repairing table` se a tabela precisar ser reparada. Se você obter vários desses erros, sem que o `mysqld` finalize inesperadamente um pouco antes, então algo está errado e precisa ser investigado melhor. Veja Seção 4.1.1 [Opções de linha de comando], Página 207.

É claro que não é um bom sinal se o `mysqld` morreu inesperadamente, mas neste caso não se deve invwestigar as mensagens `Checking table...` e sim tentar descobrir por que o `mysqld` morreu.

E.1.6 Fazendo um Caso de Teste Se Ocorre um Corrompimento de Tabela

Se você têm tabelas corrompidas ou se o `mysqld` sempre falha depois de alguns comendos de atualização, você pode testar se este erro é reproduzível fazendo o seguinte:

- Desligue o daemon MySQL (com `mysqladmin shutdown`).
- Faça um backup das tabelas (para o caso da reparação fazer algo errado)
- Verifique todas as tabelas com `myisamchk -s database/*.MYI`. Repare qualquer tabela errada com `myisamchk -r database/table.MYI`.
- Faça um segundo backup das tabelas.
- Remove (ou mova para outro local) qualquer arquivo de log antigo do diretório de dados se você precisar de mais espaço.
- Inicie o `mysqld` com `--log-bin`. Veja Seção 4.10.4 [Log binário], Página 374. Se você quiser encontrar uma consulta que provoque uma falha no `mysqld`, você deve usar `--log --log-bin`.
- Quando você obter uma tabela danificada, pare o `servidor mysql`.
- Restaure o backup.
- Reinicie o servidor `mysqld` **sem** `--log-bin`
- Re-execute os comandos com `mysqlbinlog update-log-file | mysql`. O log de atualização está salvo no diretório do banco de dados com o nome `nomemaquina-bin.#`.
- Se as tabelas forem corrompidas novamente ou você puder fazer o `mysqld` finalizar com o comando acima, vcê terá encontrado um erro reproduzível que deve ser facilmente corrigido! Envie as tabelas e o log binário por ftp para `ftp://support.mysql.com/pub/mysql/secret/` e coloque-o em nosso sistema de

erros em `http://bugs.mysql.com/`. Se você é um consumidor com suporte, você também pode enviar um email para `support@mysql.com` para alertar a equipe do MySQL sobre o problema e tê-lo corrígido o mais rápido possível..

Você também pode usar o script `mysql_find_rows` para executar algumas das instruções de atualização se você quiser estreitar o problema.

E.2 Depurando um cliente MySQL.

Para estar apto a depurar um cliente MySQL com o pacote de depuração integradom você deve configurar o MySQL com `--with-debug` ou `--with-debug=full`. Veja Seção 2.3.3 [opções de configuração], Página 98.

Antes de executar um cliente, você deve configurar a variável de ambiente `MYSQL_DEBUG`:

```
shell> MYSQL_DEBUG=d:t:O,/tmp/client.trace
shell> export MYSQL_DEBUG
```

Isto faz com que os clientes gerem um arquivo trace em '`/tmp/client.trace`'.

Se você tiver problemas com seu próprio código cliente, você deve tentar se conectar ao servidor e executar a sua consulta usando um cliente que esteja funcionando. Faça isto executando o `mysql` no modo de depuração (assumindo que você tenha compilado o MySQL com esta opção).

```
shell> mysql --debug=d:t:O,/tmp/client.trace
```

Isto irá fornecer informação útil no caso de você enviar um relatório de erro. Veja Seção 1.7.1.3 [Relatório de erros], Página 36.

Se o seu cliente falhar em algum código aparentemente 'legal', você deve verificar se o seu arquivo '`mysql.h`' incluído corresponde com o seu arquivo da biblioteca mysql. Um erro muito comum é usar um arquivo '`mysql.h`' antigo de uma instalação MySQL antiga com uma nova biblioteca MySQL.

E.3 O Pacote DBUG

O servidor MySQL e a maioria dos clientes MySQL são compilados com o pacote DBUG originalmente criado por Fred Fish. Quando se configura o MySQL para depuração, este pacote torna possível obter um arquivo trace sobre o que o programa está depurando. Veja Seção E.1.2 [Criando arquivos trace], Página 1080.

Utiliza-se o pacote de depuração chamando o programa com a opção `--debug="..."` ou `-#...`.

A maioria dos programas MySQL tem uma string de depuração padrão que será usada se você não especificar uma opção para `--debug`. O arquivo trace padrão é normalmente `/tmp/programname.trace` no Unix e `\programname.trace` no Windows.

A string de controle de depuração é uma sequência de campos separados por dois pontos como a seguir:

```
<field_1>:<field_2>:...:<field_N>
```

Cada campo consiste de um carcater de parâmetro obrigatório seguido por uma `","` e lista de modificadores separdas por vírgula opcionais:

```
flag[,modifier,modifier,...,modifier]
```

Os carcteres de parâmetros atualmente reconhecidos são:

Parâmetro	Descrição
d	Habilita a saída de macros DBUG_<N> para o estado atual. Pode ser seguido por uma lista de palavras chaves que selecionam a saída apenas para as macros DBUG com aquela palavra chave. Uma lista de palavras chaves vazia indica a saída para todas as macros.
D	Atraso depois de cada linha de saída do depurados. O argumento é o número de décimos de segundo do atraso, sujeito às capacidades da máquina. Por exemplo, `-#D,20` atrasa em dois segundos
f	Limita a depuração e/ou rastreamento, e perfilemento da lista de funções listadas. Note que uma lista nula disabilitará todas as funções. O parâmetro "d" ou "t" apropriado ainda deve ser dado, este parâmetro só limita as suas açõesse eles estiverem habilitados.
F	Identifica o nome do arquivo fonte para cada linha de saída da depuração ou rastreamento.
i	Identifica o processo com o PID ou a ID da thread para cada linha de saída da depuração ou rastreamento.
g	Habilita a modelagem. Cria um arquivo chamado 'dbugmon.out' contendo informações que poder ser usadas para moldar o programa. Pode ser seguida por uma lista de palavras chaves que selecionam a modelagem apenas para as funções naquela lista. Uma lista nula indica que todas as funções serão consideradas.
L	Identifica o número da linha do arquivo fonte para cada linha de saída da depuração ou rastreamento.
n	Exibe a profundidade de aninhamento da função atual para cada linha de saída da depuração ou rastreamento.
N	Número de cada linha de saída do dbug.
o	Redireciona o fluxo de saída do depurador para um arquivo específico. A saída padrão é stderr.
O	Igual a `o`, mas o arquivo é realmente descarregado entre cada escrita. Quando necessário o arquivo é fechado e reaberto entre cada escrita.
p	Limita as ações do depurador para um processo específico. Um processo deve ser indentificado com a macro DBUG_PROCESS e corresponder a um dos itens especificados na lista de ações do depurador que devem ocorrer.
P	Exibe o nome do processo atual para cada linha de saída da depuração ou rastreamento.
r	Quando recebe um novo estado, não herda o nível de aninhamento da função do estado anterior. Útil quando a saída é iniciada na margem esquerda.
S	Executa a função _sanity(_file_,_line_) a cada função depurada até que _sanity() retorne algo diferente de 0. (Geralmente usada com safemalloc para encontrar falhas de memória).
t	Habilita a linhas do trace de chamada/saída de funções. Pode ser seguido por uma lista (contendo apenas um modificador) dando um o nível numérico máximo de rastreamento, além do que nenhuma saída será exibida, tanto para a depuração quanto para macros trace. O padrão é uma opção de tempo de compilação.

Alguns exemplos de strings de controle do depurador que podem aparecer em uma linha de comando do shell (o "-#" é normalmente usado para introduzir uma string de controle a um aplicativo) são:

```
-#d:t
-#d:f,main,subr1:F:L:t,20
-#d,input,output,files:n
-#d:t:i:O,\\mysqld.trace
```

No MySQL, tags comuns de serem usadas (com a opção `d`) são: `enter`,`exit`,`error`,`warning`,`info` e `loop`.

E.4 Métodos de Lock

Atualmente o MySQL só suporta bloqueios de tabela para tipos `ISAM`/`MyISAM` e `HEAP`, bloqueios a nível de página para tabelas `BDB` e bloqueio a nivel de registros para tabelas `InnoDB`. Veja Seção 5.3.1 [Bloqueio interno], Página 443. Com tabelas `MyISAM` pode se misturar livremente `INSERT` e `SELECT` sem travas, se as instruções `INSERT`s não são confiltantes. (ex.: se eles são inseridos no fim da tabela em vez de preencherem espaços liberados por dados/linhas deletados).

A partir da versão 3.23.33, você pode analisar a contenção do bloqueio de tabela no seu sistema verificando as variáveis de ambiente `Table_locks_waited` e `Table_locks_immediate`.

Para decidir se você quer usar um tipo de tabela com bloqueio a nível de registro, você deverá olhar o que o aplicativo faz e o qual é o padrão de seleção/atualização dos dados.

Pontos a favor do bloqueios de registros:

- Poucos conflitos de bloqueios ao acessar registros diferentes em muitas threads.
- Poucas alterações para rollback.
- Torna possível bloquear um único registro por um longo tempo.

Contras:

- Gasta mais memória que bloqueios a nível de página ou tabela.
- É mais lento que bloqueios a nível de página ou tabela quando usado em uma grande parte da tabela, pois deve-se fazer muito mais travamentos.
- É difinitivamente muito pior que outras travas se você frequentemente utiliza `GROUP BY` em uma grande parte dos dados ou é feita uma varredura de toda a tabela.
- Com nível de bloqueios mais altos pode-se também, com mais facilidade, suportar travas de diferentes tipos para sintonizar a aplicação já que a sobreposição de bloqueio é menos perceptível que para bloqueios a nível de registro.

Bloqueios de tabela são superiores a bloqueios a nível de página / registro nos seguintes casos:

- Muitas leituras
- Leituras e atualizações em chaves restritas; é onde atualiza ou deleta-se um registro que pode ser buscado com uma leitura de chave:

```
UPDATE nome_tbl SET coluna=valor WHERE unique_key#
DELETE FROM nome_tbl WHERE unique_key=#
```

- `SELECT` combinado com `INSERT` (e muito poucas instruções `UPDATE`s e `DELETE`s).
- Muitas varreduras / `GROUP BY` em toda a tabela sem nenhuma escrita.

Outra opções além de bloqueios a nível de página / registro:

Versioning (como usamos no MySQL para inserções concorrentes) onde você pode ter um escrita e várias leituras ao mesmo tempo. Isto significa que o banco de dados/tabelas suporta diferentes viiews para os dados dependendo de quando se começa a acessá-lo. Outros nomes deste recurso são time travel, cóia na escrita ou cópia por demanda.

Cópia por demanda é em muitos casos muito melhor que bloqueio a nível de registro ou página; o piro caso, no entanto, usa muito mais memória que a usada em travamentos normais.

Em vez de se usar bloqueio de registro pode-se usar bloqueios no aplicativo (como get_lock/release_lock no MySQL). Isto só funciona em aplicaticos bem comportados.

Em muitos casos pode se fazer um palpite sobre qual tipo de bloqueio é melhor para a aplicação. mas geralmente é muito difícil dizer que um dado tipo de bloqueio é melhor que outro; tudo depende da aplicação e diferentes partes da aplicação podem exigir diferentes tipos de bloqueios.

Aqui estão algumas dicas sobre travamento no MySQL:

A maioria das aplicações web fazem diversos selects, muito poucas deleções, atualizaoes principalmente nas chaves e inserções em tabelas específicas. A configuração base do MySQL é bem sitonizada para esta finalidade.

Usuários concorrentes não são um problema se eles não misturam atualizações com seleções que precisam examinar muitas linhas na mesma tabela.

Se é misturado inserções e exclusões na mesma tabela então `INSERT DELAYED` pode ser de grande ajuda.

Pode se também usar `LOCK TABLES` para aumentar a velocidade (muitas atualizações com um travamento simples é muito mais rápida que atualizações sem travamento). Separar as coisas em tabelas diferentes também ajudará.

Se você tiver problemas de velocidade com travamento de tabelas no MySQL, você pode estar apto a resolver isto convertendo alguma de suas tabelas para tipos `InnoDB` ou `BDB`. Veja Seção 7.5 [InnoDB], Página 643. Veja Seção 7.6 [BDB], Página 698.

A seção de otimização no manual cobre diversos aspectos de como sintonizar a sua aplicação. Veja Seção 5.2.13 [Dicas], Página 440.

E.5 Comentários Sobre Threads RTS

Tentamos usar os pacotes RTS thread com o MySQL mas nos deparamos com o seguinte problema:

Eles usam um versão antiga de diversas chamadas POSIX e é muito tedioso fazer “wrappers” para todas as funções. Estamos inclinados a pensar que seria mais fácil alterar a biblioteca de threads para a especificação POSIX mais nova.

Alguns “wrappers” já estão escritos. Veja ‘`mysys/my_pthread.c`’ para maiores informações.

Pelo menos o seguinte deve ser mudado:

`pthread_get_specific` deve usar um argumento. `sigwait` deve usar dois argumentos. Diversas funções (pelo menos `pthread_cond_wait`, `pthread_cond_timedwait`) deve retornar o código do erro. Agora eles retornam -1 e configuram `errno`.

Outro problema é que threads a nivel do usuario usam o sinal `ALRM` e isto aborta diversas das funções (`read`, `write`, `open`...). O MySQL deve fazer uma nova tentativa de interrupção em todos mas não é facil de se verifica isto.

O maior problema não solucionado é o seguinte:

Para conseguir alarmes de threads alteramos '`mysys/thr_alarm.c`' para esperar entre alarmes com `pthread_cond_timedwait()`, mas isto aborta com o erro `EINTR`. Tentamos depurar a biblioteca thread para descobrirmos porque isto acontece, mas não podemos encontrar nenhuma solução fácil.

Se alguém quiser experimentar o MySQL com RTS threads sugerimos o seguinte:

- Altere as funções que o MySQL usa da biblioteca de threads para POSIX. Isto não deve levar tanto tempo.
- Compile todas as bibliotecas com `-DHAVE_rts_threads`.
- Compile `thr_alarm`.
- Se houver alguma pequena diferença na implementação, elas devem ser corrigidas alterando '`my_pthread.h`' e '`my_pthread.c`'.
- Execute `thr_alarm`. Se ele executar sem mensagens de "aviso", "erro" ou aborto, você está na trilha certa. Aqui está uma execução bem sucedidad no Solaris:

```
Main thread: 1
Thread 0 (5) started
Thread: 5  Waiting
process_alarm
Thread 1 (6) started
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 1 (1) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 2 (2) sec
Thread: 6  Simulation of no alarm needed
Thread: 6  Slept for 0 (3) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 4 (4) sec
Thread: 6  Waiting
process_alarm
thread_alarm
```

```
Thread: 5  Slept for 10 (10) sec
Thread: 5  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 5 (5) sec
Thread: 6  Waiting
process_alarm
process_alarm

...
thread_alarm
Thread: 5  Slept for 0 (1) sec
end
```

E.6 Diferença en Entre Alguns Pacotes de Threads

MySQL é muito dependente do pacote de threads usado. Assim ao escolher uma boa plataforma para o MySQL, o pacote de threads é muito importante.

Existem pelo menos três tipos de pacotes de threads:

- Threads de usuários em um processo único. A troca de threads é gerenciada com alarmes e a bilioteca thread gerencia todas as funções não segura as threads com travamento. Operações de leitura, excrita e operação são normalmente gerênciada com uma select específica da thread que troca para outra thread se a thread em execução tiver que esperar por dados. Se os pacotes de threads do usuário estão integrados com as bibliotecas padrão (threads FreeBSD e BSDI) o pacote da thread exige menos sobreposicao que pacotes de threads que têm que mapear todas as chamadas não seguras (MIT-pthreads, FSU Pthreads e RTS threads). Em alguns ambientes (SCO, por exemplo), todas as chamadas do sistema são seguras a threads e assim o mapeamento pode ser feito muito facilmente (FSU Pthreads em SCO). Desvantagem: Todas as chamadas mapeadas levam pouco tempo e é bem malicioso para tratar todas as situações. Também há, normalmente, algumas chamadas de sistema que não são tratados pelo pacote de thread (como MIT-threads e sockets). O agendamento de threads nem sempre é ótimo.
- Threads de usuários em processos separados. A troca de threads é feita pelo kernel e todos os dados são compartilhados entre threads. O pacote de thread gerência as chamadas padrão de threads para permitir o compartilhamento de dadps entre threads. LinuxThreads é usado neste método. Desvantagem: Diversos processos. A criação de thrads é lenta. Se uma thread morrer as outras normalmente travarão e você você deverá matá-las antes de reiniciar. A troca de threads também tem um custo um pouco alto.
- Threads de kernel. A troca de threads é feita pela biblioteca de thread ou pelo kernele é muito rápida. Tudo é feito em um processo, mas em alguns sistema, `ps` pode mostrar threads diferentes. Se uma thread abortar, todo o processo é abortado. A maioria das chamadas do sistema são seguras a threads e devem exigir muito pouca sobrecarga. Solaris, HP-UX, AIX e OSF/1 têm threads de kernel.

Em alguns sistemas, threads do kernel são gerenciadas threads de usuário integrads em bibkliotecas de sistemas. Nestes casos a troca de thread pode ser feita pela biblioteca de threads e o kernel não têm real conhecimento da thread.

Apêndice F Variáveis de Ambientes do MySQL

Aqui está uma lista de todas as variáveis de ambiente que são usada diretamente ou indiretamente pelo MySQL. A maioria delas também pode ser encontrada em outros lugares neste manual.

Note que qualquer opção na linha de comando sobrescreve os valores especificados nos arquivos de configuração e variáveis de ambientes, e valores nos arquivos de configuração tem preferência sobre valores em vriáveis de ambientes.

Em muitos casos é preferível utilizar um arquivo configue em vez de uma variável de ambiente para modificar o comportamento do MySQL. Veja Seção 4.1.2 [Arquivos de opções], Página 216.

Variável	Descrição
`CXX`	Defina-a em seu compilador C++ ao rodar o configure.
`CC`	Defina-a em seu compilador C ao executar configure.
`CFLAGS`	Parâmetros para o seu compilador C ao executar o configure.
`CXXFLAGS`	Parâmetros para o seu compilador C++ ao executar o configure.
`DBI_USER`	O nome do usuário padrão para Perl DBI.
`DBI_TRACE`	Usado ao rastrear o Perl DBI.
`HOME`	O caminho padrão para o arquivo de histórico do `mysql` é '`$HOME/.mysql_history`'.
`LD_RUN_PATH`	Usado para especificar onde o seu '`libmysqlclient.so`' está.
`MYSQL_DEBUG`	Opções de debug-trace ao depurar.
`MYSQL_HISTFILE`	O caminho para o arquivo de histórico do `mysql`.
`MYSQL_HOST`	Nome de máquina padrão usada pelo cliente de linha de comando `mysql`.
`MYSQL_PS1`	Prompt de comando para usar no cliente de linha de comando `mysql`. Veja Seção 4.9.2 [mysql], Página 347.
`MYSQL_PWD`	A senha padrão ao conectar ao `mysqld`. Note que o uso disto é inseguro!
`MYSQL_TCP_PORT`	A porta TCP/IP padrão.
`MYSQL_UNIX_PORT`	O socket padrão; usado para conexões `localhost`.
`PATH`	Usado pela shell para encontrar os programas MySQL.
`TMPDIR`	O diretório onde as tabelas temporárias são criadas.
`TZ`	Pode ser configurado para seu fuso horário local. Veja Seção A.4.6 [Problemas de Fuso Horário], Página 930.
`UMASK_DIR`	A máscara de criação de diretório do usuário ao se criar diretórios. Note que é feito um AND com `UMASK`!
`UMASK`	A máscara de criação dos arquivos do usuário, usado ao se criar um arquivo.
`USER`	O usuário padrão no Windows para usar ao conectar ao `mysqld`.

Apêndice G Sintaxe de Expressões Regulares do MySQL

Um expressão regular (regex) é um modo poderoso de especificar um pesquisa complexa.

O MySQL usa a implementação do Henry Spencer de expressões regulares, a qual está em conformidade com o POSIX 1003.2. MySQL usa a versão extendida.

Esta é uma referência simplória que salta os detalhes. Para obter informações exatas, veja a página manual do `regex(7)` de Henry Spencer que está incluída na distribuicão fonte. Veja Apêndice C [Credits], Página 945.

Uma expressão regular descreve um conjunto de strings. A regexp mais simples é uma que não tenha nenhum caracter especial nela. Por exeplo, o regexp `hello` combina com `hello` e nada mais.

Expressões regulares não triviais usam certas construções especiais e assim podem encontrar mais de uma string. Por exemplo, o regexp `hello|word` combina tanto com a string `hello` quanto com a string `word`.

Como um exemplo mais complicado, o regexp `B[an]*s` mcombina com qualquer das strings `Bananas`, `Baaaaas`, `Bs`, e qualquer string iniciando com um `B`, e finalizando com um `s`, e contendo qualquer número de caracteres `a` ou `n` entre eles.

Um expressão reguklar pode utilizar qualquer dos um dos caracteres/ construtores especiais:

`^` Combina com o inicio de uma string.

```
mysql> SELECT "fo\nfo" REGEXP "^fo$";            -> 0
mysql> SELECT "fofo" REGEXP "^fo";               -> 1
```

`$` Combina com o fim de uma string.

```
mysql> SELECT "fo\no" REGEXP "^fo\no$";          -> 1
mysql> SELECT "fo\no" REGEXP "^fo$";             -> 0
```

`.` Combina com qualquer caracter (incluindo novas linhas)

```
mysql> SELECT "fofo" REGEXP "^f.*";              -> 1
mysql> SELECT "fo\nfo" REGEXP "^f.*";            -> 1
```

`a*` Combina com qualquer sequência de zero ou mais carcteres `a`.

```
mysql> SELECT "Ban" REGEXP "^Ba*n";              -> 1
mysql> SELECT "Baaan" REGEXP "^Ba*n";            -> 1
mysql> SELECT "Bn" REGEXP "^Ba*n";               -> 1
```

`a+` Cobina com qualquer sequência de um ou mais caracteres `a`.

```
mysql> SELECT "Ban" REGEXP "^Ba+n";              -> 1
mysql> SELECT "Bn" REGEXP "^Ba+n";               -> 0
```

`a?` Combina com zero ou um caracter `a`.

```
mysql> SELECT "Bn" REGEXP "^Ba?n";               -> 1
mysql> SELECT "Ban" REGEXP "^Ba?n";              -> 1
mysql> SELECT "Baan" REGEXP "^Ba?n";             -> 0
```

`de|abc` Combina tant com a sequencia `de` como com `abc`.

```
mysql> SELECT "pi" REGEXP "pi|apa";           -> 1
mysql> SELECT "axe" REGEXP "pi|apa";          -> 0
mysql> SELECT "apa" REGEXP "pi|apa";          -> 1
mysql> SELECT "apa" REGEXP "^(pi|apa)$";      -> 1
mysql> SELECT "pi" REGEXP "^(pi|apa)$";       -> 1
mysql> SELECT "pix" REGEXP "^(pi|apa)$";      -> 0
```

`(abc)*` Combina com zero ou mais instâncias da sequência `abc`.

```
mysql> SELECT "pi" REGEXP "^(pi)*$";          -> 1
mysql> SELECT "pip" REGEXP "^(pi)*$";         -> 0
mysql> SELECT "pipi" REGEXP "^(pi)*$";        -> 1
```

`{1}`
`{2,3}` Existe um modo mais geral de se escrever regexp que combinam com muitas ocorrências de um átomo anterior.

`a*` Pode ser escrito como `a{0,}`.

`a+` Pode ser escrito como `a{1,}`.

`a?` Pode ser escrito como `a{0,1}`.

Para ser mais preciso, um átomo seguido por um limite contendo um inteiro `i` e nenhuma vírgula casa com uma sequência de exatamente `i` combinações do átomo. Um átomo seguido por um limite contendo `i` e uma virgula casa com uma sequência de `i` ou mais combinações do átomo. Um átomo seguido por um limite contendo dois inteiros `i` e `j` casa com uma seqquência de `i` até `j` (inclusive) combinações de átomos.

Ambos os argumentos devem estar na faixa de `0` ate `RE_DUP_MAX` (padrão é 255), inclusive. Se houver dois argumentos, o segundo deve ser maior ou igual ao primeiro.

`[a-dX]`
`[^a-dX]` Combina com qualquer caracter que seja (ou não, se `^` é usado) `a`, `b`, `c`, `d` ou `X`. Para incluir um caracter literal `]`, ele deve ser imediatamente seguido pelo colchete de abertura `[`. Para incluir um caracter literal `-`, ele deve ser escrito primeiro ou por ultimo. Assim o `[0-9]` encontra qualquer dígito decimal. Qualquer caracter que não tenha um significado definido dentro de um para `[]` não tem nenhum significado especial e combina apenas com ele mesmo.

```
mysql> SELECT "aXbc" REGEXP "[a-dXYZ]";       -> 1
mysql> SELECT "aXbc" REGEXP "^[a-dXYZ]$";     -> 0
mysql> SELECT "aXbc" REGEXP "^[a-dXYZ]+$";    -> 1
mysql> SELECT "aXbc" REGEXP "^[^a-dXYZ]+$";   -> 0
mysql> SELECT "gheis" REGEXP "^[^a-dXYZ]+$";  -> 1
mysql> SELECT "gheisa" REGEXP "^[^a-dXYZ]+$"; -> 0
```

`[[.caracter.]]`
A sequência de caracteres daquele elemento ordenado. A sequência é um único elemento da lista de expressões entre colchetes. Um expressão entre colchetes contendo um elemento ordenado multi-caracter pode então combinar com mais

de um caracter, por exemplo, se a sequência ordenada inclui um elemento ordenado `ch`, então a expresão regular `[[.ch.]]*c` casa com os primeiros cinco caracteres de `chchcc`.

`[=classe_caracter=]`

Uma classe equivalente, procura pela sequência de caracteres de todos elementos ordenados equivalentes àquele, incluindo ele mesmo.

Por exemplo, se `o` e `(+)` são os membros de uma classe equivalente, então `[[=o=]]`, `[[=(+)=]]` e `[o(+)]` são todos sinônimos. Uma classe equivalente não pode ser o final de uma escala.

`[:character_class:]`

Dentro de colchets, o nome de uma classe de caracter entre `[:` e `:]` procura pela lista de todos os caracteres pertencentes a esta classe. Os nomes de classes de caracteres padrões são:

Nome	**Nome**	**Nome**
alnum	digit	punct
alpha	graph	space
blank	lower	upper
cntrl	print	xdigit

Ele procura pelas classes de caracteres definidas na página `ctype(3)` do manual. Um local pode forncer outros. Uma classe de caracter não pode ser usada como o final de uma escala.

```
mysql> SELECT "justalnums" REGEXP "[[:alnum:]]+";       -> 1
mysql> SELECT "!!" REGEXP "[[:alnum:]]+";               -> 0
```

`[[:<:]]`

`[[:>:]]` Combina com a string null no inicio e no fim de uma palavra, respectivamente. Uma palavra é definida como uma sequencia de caracteres de palavra os quais não são nem precedido e nem seguidos por caracteres de palavras. Um caracter de palavra é um caracter alfa numérico (como definido por `ctype(3)`) ou um underscore (_).

```
mysql> SELECT "a word a" REGEXP "[[:<:]]word[[:>:]]";      -> 1
mysql> SELECT "a xword a" REGEXP "[[:<:]]word[[:>:]]";     -> 0
```

```
mysql> SELECT "weeknights" REGEXP "^(wee|week)(knights|nights)$"; -> 1
```

Apêndice H GPL - Licença Pública Geral do GNU

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions

for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. Accompany it with a written offer, valid for at least three years, to give any third-party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.
4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you

indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy  name of author

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Índices dos Comandos, Tipos e Funções SQL

!

! (NOT logico) 508
!= (diferente) 506

"

" 472

%

% (meta caracter) 469
% (modulo) 523

&

& (operado binário AND) 546
&& (AND lógico) 509

(

() (parenteses) 504
(Control-Z) \z 469

*

* (multiplicação) 522

+

+ (adição) 522

-

- (menos unário) 523
- (subtração) 522
-p opções 267
-password opções 267

.

.my.cnf file 226
.mysql_history, arquivo 345
.pid (process ID), arquivo 292

/

/ (divisão) 522
‘/etc/passwd’ 230
/etc/passwd 566

<

< (menor que) 506
<< 201
<< (deslocamento a esquerda) 546
<= (menor que ou igual) 506
<=> (Igual a) 506
<> (diferente) 506

=

= (igual) 505

>

> (maior que) 506
>= (maior que ou igual) 506
>> (deslocamento a direita) 546

^

^ (operado binário XOR) 546

_

_ (meta caracter) 469

‘

‘ 472

\

\" (aspas duplas) 469
\’ (aspas simples) 469
\\ (escape) 469
\0 (ASCII 0) 469
\b (backspace) 469
\n (nova linha) 469
\r (retorno de carro) 469
\t (tab) 469
\z (Control-Z) ASCII(26) 469

|

| (operador binário OR) 546
|| (OR lógico) 509

~

~ 546

A

ABS() ... 523
ACOS() ... 526
ADDDATE() ... 536
ADDTIME() ... 536
adição (+) ... 522
AES_DECRYPT() ... 549
AES_ENCRYPT() ... 549
alias ... 933
ALTER COLUMN ... 610
ALTER FUNCTION ... 766
ALTER PROCEDURE ... 766
ALTER TABLE ... 608, 610, 938
ANALYZE TABLE ... 298
AND lógico ... 509
AND, operado binário ... 546
Area() ... 753, 754
aritiméticas, funções ... 546
ARQUIVO ... 515
arquivo .my.cnf ... 135, 216, 218, 239, 253, 267
AS ... 563, 567
AsBinary() ... 748
ASCII() ... 512
ASIN() ... 526
aspas duplas (\") ... 469
aspas em identificadores ... 472
aspas simples (\') ... 469
AsText() ... 748
ATAN() ... 526
ATAN2() ... 526
AUTO_INCREMENT, usando com DBI ... 886
AVG() ... 556

B

backspace (\b) ... 469
BACKUP TABLE ... 277
BdMPolyFromText() ... 743
BdMPolyFromWKB() ... 744
BdPolyFromText() ... 743
BdPolyFromWKB() ... 744
BEGIN ... 615, 767
BENCHMARK() ... 553
BETWEEN ... AND ... 507
BIGINT ... 482
BIN() ... 512
binárias, funções ... 546
BINARY ... 521
BIT ... 482, 486
BIT_AND() ... 558
BIT_COUNT ... 201
BIT_COUNT() ... 546
BIT_LENGTH() ... 512
BIT_OR ... 201
BIT_OR() ... 558
BIT_XOR() ... 558
BLOB ... 486, 498
BOOL ... 482, 486
BOOLEAN ... 482
Boundary() ... 750
Buffer() ... 755

C

C:\my.cnf file ... 226
CALL ... 767
CASE ... 510, 772
CAST ... 544
CC variáveis de ambiente ... 98
CC, variável de ambiente ... 105, 1092
CEILING() ... 524
Centroid() ... 754
CFLAGS, variável de ambiente ... 105, 1092
CHANGE MASTER TO ... 402
CHAR ... 485, 496
CHAR VARYING ... 486
CHAR() ... 512
CHAR_LENGTH() ... 516
CHARACTER ... 485
CHARACTER VARYING ... 486
CHARACTER_LENGTH() ... 516
CHECK TABLE ... 278
CHECKSUM TABLE ... 299
ChopBlanks, método DBI ... 885
CLOSE ... 771
COALESCE() ... 508
coerção ... 521
Comentário sintaxe ... 479
COMMIT ... 46, 615
comparação, operadores de ... 504
COMPRESS() ... 550
CONCAT() ... 512
CONCAT_WS() ... 512
connect(), método DBI ... 882
CONNECTION_ID() ... 552
Contains() ... 757
CONV() ... 513
CONVERT ... 544
ConvexHull() ... 755
COS() ... 526
COT() ... 527
COUNT() ... 556
COUNT(DISTINCT) ... 556
CRC32() ... 527
CREATE DATABASE ... 597
CREATE FUNCTION ... 764, 900
CREATE INDEX ... 613
CREATE PROCEDURE ... 764
CREATE TABLE ... 598
CROSS JOIN ... 567
Crosses() ... 757
CSS, variavel de ambiente ... 104
CURDATE() ... 541
CURRENT_DATE ... 541
CURRENT_TIME ... 541
CURRENT_TIMESTAMP ... 541

CURRENT_USER() 547
CURTIME() 541
CXX varáveis de ambiente 98
CXX, variável de ambiente 105, 1092
CXXFLAGS environment variable 99
CXXFLAGS, varáveis de ambiente 99
CXXFLAGS, variavel de ambiente 105
CXXFLAGS, variável de ambiente 1092

D

data e hora, funções de 529
data_sources(), método DBI 885
Database information, obtaining 302
DATABASE() 547
DATE 484, 491, 931
DATE() 530
DATE_ADD() 534
DATE_FORMAT() 537
DATE_SUB() 534
DATEDIFF() 536
DATETIME 484, 491
DAY() 530
DAYNAME() 531
DAYOFMONTH() 530
DAYOFWEEK() 530
DAYOFYEAR() 530
DBI->{ChopBlanks} 885
DBI->{is_blob} 886
DBI->{is_key} 886
DBI->{is_not_null} 887
DBI->{is_num} 886
DBI->{is_pri_key} 886
DBI->{length} 887
DBI->{max_length} 887
DBI->{mysql_insertid} 886
DBI->{NAME} 887
DBI->{NULLABLE} 885
DBI->{NUM_OF_FIELDS} 885
DBI->{table} 887
DBI->{type} 887
DBI->connect() 882
DBI->data_sources() 885
DBI->disconnect 883
DBI->do() 884
DBI->execute 884
DBI->fetchall_arrayref 885
DBI->fetchrow_array 884
DBI->fetchrow_arrayref 884
DBI->fetchrow_hashref 884
DBI->finish 885
DBI->prepare() 883
DBI->quote 470
DBI->quote() 884
DBI->rows 885
DBI->trace 885, 1082
DBI_TRACE, variável de ambiente 886
DBI_TRACE, variável de ambiente 1082, 1092
DBI_USER, variável de ambiente 1092
DEC 484
DECIMAL 484
DECLARE 767
DECODE() 548
DEGREES() 528
DELAYED 582
DELETE 585
DES_DECRYPT() 550
DES_ENCRYPT() 549
DESC 614
DESCRIBE 192, 614
diferente (!=) 506
diferente (<>) 506
Difference() 755
Dimension() 749
disconnect, método DBI 883
Disjoint() 757
Distance() 758
DISTINCT 180, 434, 556
DIV 524
diversas, funções 547
divisão (/) 522
DO 597
do(), método DBI 884
DOUBLE 484
DOUBLE PRECISION 484
DROP DATABASE 597
DROP FUNCTION 766, 900
DROP INDEX 610, 614
DROP PRIMARY KEY 610
DROP PROCEDURE 766
DROP TABLE 612
DROP USER 264
DUMPFILE 567

E

ELT() 513
ENCODE() 548
ENCRYPT() 548
END 767
EndPoint() 751
ENUM 487, 499
Envelope() 749
environment variable, CXXFLAGS 99
Environment variable, LD_RUN_PATH 166
Equals() 757
escape (\\) 469
execute, método DBI 884
EXP() 524
EXPLAIN 424
EXPORT_SET() 513
ExteriorRing() 753
EXTRACT() 536

F

FALSE ... 471
FETCH ... 771
fetchall_arrayref, método DBI ... 885
fetchrow_array, método DBI ... 884
fetchrow_arrayref, método DBI ... 884
fetchrow_hashref, método DBI ... 884
FIELD() ... 513
FILE ... 515
FIND_IN_SET() ... 514
finish, método DBI ... 885
FIXED ... 484
FLOAT ... 483
FLOAT(M,D) ... 483
FLOAT(precisão) ... 483, 484
FLOOR() ... 523
FLUSH ... 299
fluxo de controle, funções ... 510
FORCE INDEX ... 563, 568
FORMAT() ... 552
FOUND_ROWS() ... 554
FROM ... 563
FROM_DAYS() ... 537
FROM_UNIXTIME() ... 542
funções de data e hora ... 529
funções de fluxo de controle ... 510
funções definidas pelo usuário ... 900
funções matematicas ... 523
funções string ... 512
funções, aritiméticas ... 546
funções, binárias ... 546
funções, diversas ... 547
funções, GROUP BY ... 556
functions de comparação de string ... 519

G

GeomCollFromText() ... 743
GeomCollFromWKB() ... 744
GEOMETRY ... 742
GEOMETRYCOLLECTION ... 742
GeometryCollection() ... 745
GeometryCollectionFromText() ... 743
GeometryCollectionFromWKB() ... 744
GeometryFromText() ... 742
GeometryFromWKB() ... 743
GeometryN() ... 754
GeometryType() ... 749
GeomFromText() ... 742, 748
GeomFromWKB() ... 743, 748
GET_FORMAT() ... 539
GET_LOCK() ... 553
GLength() ... 751, 752
GRANT ... 254
GRANT, instrução ... 273
GRANT, instruções ... 261
GRANTS ... 322
GREATEST() ... 528
GROUP BY, funções ... 556
GROUP_CONCAT() ... 557

H

HANDLER ... 595
HEX() ... 514
HOME variável de ambiente ... 345
HOME, variável de ambiente ... 1092
host.frm, problemas encontrando ... 112
HOUR() ... 533

I

identificadores, aspas ... 472
IF ... 771
IF() ... 510
IFNULL() ... 511
IGNORE INDEX ... 563, 568
IGNORE KEY ... 563, 569
igual (=) ... 505
IN ... 507
INET_ATON() ... 554
INET_NTOA() ... 554
INNER JOIN ... 567
INSERT ... 438, 579
INSERT ... SELECT ... 581
INSERT DELAYED ... 582
INSERT() ... 514
INSERT, cooncessão de privilégios com instruções ... 262
INSTR() ... 514
instruções GRANT ... 261
instruções INSERT ... 262
INT ... 482
INTEGER ... 482
InteriorRingN() ... 753
Intersection() ... 755
Intersects() ... 757
INTERVAL() ... 508
IS NOT NULL ... 506
IS NULL ... 434, 506
IS NULL, e índices ... 449
is_blob, método DBI ... 886
IS_FREE_LOCK() ... 553
is_key, método DBI ... 886
is_not_null, método DBI ... 887
is_num, método DBI ... 886
is_pri_key, método DBI ... 886
IsClosed() ... 751, 752
IsEmpty() ... 750
ISNULL() ... 507
ISOLATION LEVEL ... 619
IsRing() ... 752
IsSimple() ... 750
ITERATE ... 773

J

JOIN 567

K

KILL 301

L

LAST_DAY() 540
LAST_INSERT_ID() 49
LAST_INSERT_ID([expr]) 551
LCASE() 515
LD_RUN_PATH environment variable 166
LD_RUN_PATH variáveis de ambiente 147
LD_RUN_PATH, variável de ambiente ... 140, 1092
LEAST() 528
LEAVE 772
LEFT JOIN 435, 567
LEFT OUTER JOIN 567
LEFT() 515
LENGTH() 516
length, método DBI 887
LIKE 519
LIKE, e índices 448
LIKE, e meta caracteres 448
LIMIT 437, 554
LineFromText() 743
LineFromWKB() 744
LINESTRING 742
LineString() 745
LineStringFromText() 743
LineStringFromWKB() 744
LN() 524
LOAD DATA FROM MASTER 404
LOAD DATA INFILE 588, 932
LOAD TABLE FROM MASTER 405
LOAD_FILE() 515
LOCALTIME 541
LOCALTIMESTAMP 541
LOCATE() 515
LOCK TABLES 617
LOG() 525
LOG10() 525
LOG2() 525
logico, operadores 508
LONG 498
LONGBLOB 486
LONGTEXT 486
LOOP 772
LOWER() 515
LTRIM() 515

M

maior que (>) 506
maior que ou igual (>=) 506
MAKE_SET() 516
MAKEDATE() 541
MAKETIME() 541
MASTER_POS_WAIT() 405, 554
MATCH ... AGAINST() 521
matematicas, funções 523
MAX() 556
max_length, método DBI 887
MBRContains() 756
MBRDisjoint() 756
MBREqual() 756
MBRIntersects() 756
MBROverlaps() 757
MBRTouches() 757
MBRWithin() 756
MD5() 548
MEDIUMBLOB 486
MEDIUMINT 482
MEDIUMTEXT 486
menor que (<) 506
menor que ou igual (<=) 506
menos unário (-) 523
Meta caracter (%) 469
Meta caracter (_) 469
MICROSECOND() 533
MID() 516
MIN() 556
MINUTE() 533
MLineFromText() 743
MOD (modulo) 523
MOD() 523
modulo (%) 523
modulo (MOD) 523
MONTH() 531
MONTHNAME() 531
MPointFromText() 743
MPointFromWKB() 744
MPolyFromText() 743
MPolyFromWKB() 744
MULTILINESTRING 742
MultiLineString() 745
MultiLineStringFromText() 743
multiplicação (*) 522
MULTIPOINT 742
MultiPoint() 745
MultiPointFromText() 743
MultiPointFromWKB() 744
MULTIPOLYGON 742
MultiPolygon() 745
MultiPolygonFromText() 743
MultiPolygonFromWKB() 744
my_init() 858
my_ulonglong, imprimindo valores 776
my_ulonglong, tipo C 776
MySQL C type 830

MYSQL, tipo C 776
mysql_affected_rows() 783
mysql_autocommit() 826
MYSQL_BIND, tipo C 828
mysql_bind_param() 836
mysql_bind_result() 842
mysql_change_user() 784
mysql_character_set_name() 785
mysql_close() 786
mysql_commit() 825
mysql_connect() 786
mysql_create_db() 787
mysql_data_seek() 787
MYSQL_DEBUG variável de ambiente 345
MYSQL_DEBUG variável de ambiente 1085
mysql_debug() 788
MYSQL_DEBUG, variável de ambiente 1092
mysql_drop_db() 788
mysql_dump_debug_info() 789
mysql_eof() 790
mysql_errno() 791
mysql_error() 792
mysql_escape_string() 470
mysql_escape_string() 793
mysql_execute() 837
mysql_fetch() 846
mysql_fetch_field() 793
mysql_fetch_field_direct() 794
mysql_fetch_fields() 794
mysql_fetch_lengths() 795
mysql_fetch_row() 796
MYSQL_FIELD, tipo C 776
mysql_field_count() 797, 807
MYSQL_FIELD_OFFSET, tipo C 776
mysql_field_seek() 798
mysql_field_tell() 799
mysql_free_result() 799
mysql_get_client_info() 800
mysql_get_client_version() 800
mysql_get_host_info() 800
mysql_get_metadata 835
mysql_get_proto_info() 801
mysql_get_server_info() 801
mysql_get_server_version() 801
MYSQL_HISTFILE variável de ambiente 345
MYSQL_HISTFILE, variável de ambiente 1092
MYSQL_HOST, variável de ambiemte 239
MYSQL_HOST, variável de ambiente 1092
mysql_info() 581, 584, 595, 610
mysql_info() 802
mysql_init() 803
mysql_insert_id() 49
mysql_insert_id() 803
mysql_insertid DBI, atrinuto 886
mysql_kill() 804
mysql_list_dbs() 804
mysql_list_fields() 805
mysql_list_processes() 806
mysql_list_tables() 806
mysql_more_results() 826
mysql_next_result() 827
mysql_num_fields() 807
mysql_num_rows() 808
mysql_options() 809
mysql_param_count() 835
mysql_ping() 811
mysql_prepare() 834
MYSQL_PS1, variável de ambiente 1092
MYSQL_PWD variável de ambiente 345
MYSQL_PWD, variável de ambiente 239
MYSQL_PWD, variável de ambiente 1092
mysql_query() 812, 861
mysql_real_connect() 813
mysql_real_escape_string() 815
mysql_real_query() 816
mysql_reload() 817
MYSQL_RES, tipo C 776
mysql_rollback() 826
MYSQL_ROW, tipo C 776
mysql_row_seek() 818
mysql_row_tell() 818
mysql_select_db() 819
mysql_send_long_data() 851
mysql_server_end() 860
mysql_server_init() 859
mysql_set_sever_option() 819
mysql_shutdown() 820
mysql_sqlstate() 820
mysql_ssl_set() 821
mysql_stat() 822
MYSQL_STMT, tipo C 828
mysql_stmt_affected_rows() 842
mysql_stmt_close() 853
mysql_stmt_data_seek() 844
mysql_stmt_errno() 854
mysql_stmt_error() 854
mysql_stmt_num_rows() 845
mysql_stmt_row_seek() 844
mysql_stmt_row_tell() 845
mysql_stmt_sqlstate() 855
mysql_stmt_store_result() 843
mysql_store_result() 822, 861
MYSQL_TCP_PORT variável de ambiente 345
MYSQL_TCP_PORT, variavel de ambiente 225
MYSQL_TCP_PORT, variável de ambiente 1092
mysql_thread_end() 858
mysql_thread_id() 823
mysql_thread_init() 858
mysql_thread_safe() 859
MYSQL_UNIX_PORT variável de ambiente 345
MYSQL_UNIX_PORT, variável de ambiente 116
MYSQL_UNIX_PORT, variavel de ambiente 225
MYSQL_UNIX_PORT, variável de ambiente 1092
mysql_use_result() 824
mysql_warning_count() 825

N

NAME, método DBI ... 887
NATIONAL CHAR ... 485
NATURAL LEFT JOIN ... 567
NATURAL LEFT OUTER JOIN ... 567
NATURAL RIGHT JOIN ... 567
NATURAL RIGHT OUTER JOIN ... 567
NCHAR ... 485
NOT BETWEEN ... 507
NOT IN ... 507
NOT LIKE ... 520
NOT logico ... 508
NOT REGEXP ... 521
nova linha (\n) ... 469
NOW() ... 541
NUL ... 469
NULL ... 185, 932
NULL, valor ... 472
NULLABLE, método DBI ... 885
NULLIF() ... 511
NUM_OF_FIELDS, método DBI ... 885
NUMERIC ... 484
NumGeometries() ... 754
NumInteriorRings() ... 753
NumPoints() ... 751

O

OCT() ... 516
OCTET_LENGTH() ... 516
OLD_PASSWORD() ... 547
opções de configuração, --with-charset ... 99
opções de configuração, --with-extra-charset ... 99
opções de linha de comando ... 207
OPEN ... 771
operadores logicos ... 508
OPTIMIZE TABLE ... 298
OR logico ... 509
OR, operador binário ... 546
ORD() ... 516
ORDER BY ... 610
Overlaps() ... 757

P

parenteses (e) ... 504
PASSWORD() ... 240, 266, 547, 920
PATH, variavel de ambiente ... 93
PATH, variável de ambiente ... 1092
PERIOD_ADD() ... 533
PERIOD_DIFF() ... 533
PI() ... 526
POINT ... 742
Point() ... 745
PointFromText() ... 742
PointFromWKB() ... 744
PointN() ... 752
PointOnSurface() ... 754
PolyFromText() ... 743
PolyFromWKB() ... 744
POLYGON ... 742
Polygon() ... 745
PolygonFromText() ... 743
PolygonFromWKB() ... 744
POSITION() ... 517
POW() ... 525
POWER() ... 525
prepare(), método DBI ... 883
PRIMARY KEY ... 602, 610
PROCESSLIST ... 320
PURGE MASTER LOGS ... 400

Q

QUARTER() ... 531
QUOTE() ... 517
quote(), método DBI ... 884

R

RADIANS() ... 528
RAND() ... 527
REAL ... 484
ref_or_null ... 434
REGEXP ... 520
Related() ... 758
RELEASE_LOCK() ... 553
RENAME TABLE ... 612
REPAIR TABLE ... 279
REPEAT ... 773
REPEAT() ... 517
REPLACE ... 587
REPLACE ... SELECT ... 581
REPLACE() ... 517
REQUIRE GRANT, opções ... 273
RESET MASTER ... 401
RESET SLAVE ... 405
RESTORE TABLE ... 277
restrições ... 52
retorno (\r) ... 469
retorno de carro (\r) ... 469
REVERSE() ... 517
REVOKE ... 254
RIGHT JOIN ... 567
RIGHT OUTER JOIN ... 567
RIGHT() ... 517
RLIKE ... 520
ROLLBACK ... 46, 615
ROLLBACK TO SAVEPOINT ... 616
ROLLUP ... 558
ROUND() ... 524
rows, método DBI ... 885
RPAD() ... 517
RTRIM() ... 517

S

SAVEPOINT 616
SEC_TO_TIME() 543
SECOND() 533
SELECT 562
SELECT INTO 768
SELECT INTO TABLE 46
SELECT, otimizando 424
SELECT, velocidade do 431
SESSION_USER() 547
SET 460, 487, 500, 768
SET GLOBAL SQL_SLAVE_SKIP_COUNTER 405
SET OPTION 460
SET PASSWORD, instruções 266
SET SQL_LOG_BIN 401
SET TRANSACTION 619
SHA() 548
SHA1() 548
SHOW BINLOG EVENTS 401
SHOW COLUMNS 302
SHOW CREATE FUNCTION 767
SHOW CREATE PROCEDURE 767
SHOW CREATE TABLE 302, 322
SHOW DATABASES 302
SHOW FIELDS 302
SHOW FUNCTION STATUS 767
SHOW GRANTS 302, 322
SHOW INDEX 302
SHOW KEYS 302
SHOW MASTER LOGS 302, 401
SHOW MASTER STATUS 302, 401
SHOW PRIVILEGES 325
SHOW PROCEDURE STATUS 767
SHOW PROCESSLIST 302, 320
SHOW SLAVE HOSTS 401
SHOW SLAVE STATUS 302, 406
SHOW STATUS 302
SHOW TABLE STATUS 302
SHOW TABLE TYPES 302, 324
SHOW TABLES 302
SHOW VARIABLES 302
SHOW WARNINGS 302, 323
SIGN() 523
SIN() 526
SMALLINT 482
SOUNDEX() 518
SOUNDS LIKE 520
SPACE() 518
SQL_CACHE 628
SQL_NO_CACHE 628
SQRT() 525
SRID() 749
START SLAVE 409
START TRANSACTION 615
StartPoint() 752
STD() 558
STDDEV() 558
STOP SLAVE 410
STR_TO_DATE() 539
STRAIGHT_JOIN 567
STRCMP() 521
string, funções 512
string, funções de comparação de 519
SUBDATE() 540
SUBSTRING() 518
SUBSTRING_INDEX() 518
SUBTIME() 540
subtração (-) 522
SUM() 557
SymDifference() 755
SYSDATE() 541
SYSTEM_USER() 547

T

tab (\t) 469
Table scan 937
table, método DBI 887
table_cache 451
TAN() 526
TEXT 486, 498
threads 320
TIME 485, 495
TIME() 530
TIME_FORMAT() 540
TIME_TO_SEC() 543
TIMEDIFF() 536
TIMESTAMP 484, 491
TIMESTAMP() 530
TINYBLOB 486
TINYINT 482
TINYTEXT 486
Tipos 482
TMPDIR, variável de ambiente 116, 1092
TO_DAYS() 537
Touches() 757
trace DBI, método 885
trace, método DBI 1082
TRIM() 519
TRUE 471
TRUNCATE 587
TRUNCATE() 528
type, método DBI 887
TZ, variável de ambiente 930, 1092

U

UCASE() 519
UDF, funções 900
ulimit 922
UMASK, variavel de ambiente 924
UMASK, variável de ambiente 1092
UMASK_DIR, variável de ambiente 924, 1092
unário, menos (-) 523
UNCOMPRESS() 551
UNCOMPRESSED_LENGTH() 551

união (UNION) 200
UNION 569
Union() 755
UNIQUE 610
UNIX_TIMESTAMP() 542
UNLOCK TABLES 617
UNTIL 773
UPDATE 584
UPPER() 519
USE 614
USE INDEX 563, 568
USE KEY 563, 569
USER() 547
USER, variável de ambiente 239
USER, variável de ambiente 1092
usuário, funções definidas por 900
UTC_DATE() 543
UTC_TIME() 543
UTC_TIMESTAMP() 543

V

vairaveis de ambiente CXXFLAGS 99
valores hexadecimais 471
VARCHAR 486, 496
VARCHARACTER 486
VARIANCE() 557
variáveis de ambiente, CC 98
variáveis de ambiente, CXX 98
Variáveis de ambiente, LD_RUN_PATH 147
variável de ambiente CC 105
Variável de ambiente CC 1092
variável de ambiente CFLAGS 105
Variável de ambiente CFLAGS 1092
variavel de ambiente CXX 104
variável de ambiente CXX 105
Variável de ambiente CXX 1092
variável de ambiente CXXFLAGS 105
Variável de ambiente CXXFLAGS 1092
variável de ambiente DBI_TRACE 886
Variável de ambiente DBI_TRACE 1092
Variável de ambiente DBI_USER 1092
Variável de ambiente HOME 1092
variável de ambiente LD_RUN_PATH 140
Variável de ambiente LD_RUN_PATH 1092
Variável de ambiente MYSQL_DEBUG 1092
Variável de ambiente MYSQL_HISTFILE 1092
variável de ambiente MYSQL_HOST 239
Variável de ambiente MYSQL_HOST 1092
Variável de ambiente MYSQL_PS1 1092
variável de ambiente MYSQL_PWD 239
Variável de ambiente MYSQL_PWD 1092
variável de ambiente MYSQL_TCP_PORT 225
Variável de ambiente MYSQL_TCP_PORT 1092
variável de ambiente MYSQL_UNIX_PORT 225
Variável de ambiente MYSQL_UNIX_PORT 116, 1092
Variável de ambiente PATH 1092
Variável de ambiente TMPDIR 116, 1092
Variável de ambiente TZ 930, 1092
Variável de ambiente UMASK 924, 1092
Variável de ambiente UMASK_DIR 924, 1092
variável de ambiente USER 239
Variável de ambiente USER 1092
variável de ambiente, DBI_TRACE 1082
variável de ambiente, HOME 345
variável de ambiente, MYSQL_DEBUG 345
variável de ambiente, MYSQL_DEBUG 1085
variável de ambiente, MYSQL_HISTFILE 345
variável de ambiente, MYSQL_PWD 345
variavel de ambiente, MYSQL_TCP_PORT 225
variável de ambiente, MYSQL_TCP_PORT 345
variavel de ambiente, MYSQL_UNIX_PORT 225
variável de ambiente, MYSQL_UNIX_PORT 345
variável de ambiente, PATH 93
VERSION() 552

W

WEEK() 531
WEEKDAY() 530
WEEKOFYEAR() 532
WHERE 432
WHILE 774
Within() 758
without-server opção 98

X

X() 750
XOR, logical 509
XOR, operado binário 546

Y

Y() 750
YEAR 485, 496
YEAR() 532

Concept Index

-

`--with-raid`, erros de ligação 104

A

abertas, tables 451
abortados, clientes 918
abortados, conexão 918
abrindo tabelas 451
access denied 912
accesso negado, erro 912
acesso, controle 239
acesso, privilégio 226
`ACID` 46
ACID 643
ACLs 226
ActiveState Perl 165
adicionando conjunto de caracteres 327
adicionando funções nativas 908
adicionando novas funções 899
adicionando novos privilégios de usuários 261
adicionando novos usuários 92
adicionando, funções definidas por usuário 900
adicionando, procedimentos 909
administração de servidor 356
ADO program 875
agrupando, expressões 504
aliases, em cláusulas `GROUP BY` 562
aliases, em cláusulas `ORDER BY` 562
aliases, em expressões 563
aliases, nomes 472
aliases, para expressões 562
aliases, para tabelas 564
alterações de colunas sem aviso 607
alterações na versão 4.0 966
alterações na versão 4.1 957
alterações, log 957
alterações, versão 5.0 957
alterando a localização do socket 119, 929
alterando a ordem das colunas 938
alterando a tabela 610
alterando campos 610
alterando localização do socket 98
alterando tabelas 608, 938
alterando, colunas 610
Ano 2000, assuntos referentes ao 490
anônimo, usuário 240, 260
anonymous user 260
anoônimo, usuário 243
ANSI modo, executando 42
Apache 206
apagando, usuário 264
API C, funções 778
API C, problemas com ligação 862
API's, lista de 954
APIs 775
APIs, Perl 881
aplicando, patches 97
argumentos, processando 904
aritiméticas, expressões 522
armazenamento de dados 446
armazenamento dos tipos de colunas, exigências 502
arquivo `config.cache` 103
arquivo de opções 216
arquivo não encontrado, mensagem 924
arquivo RPM 69
arquivo temporário, direito de escrita 116
arquivo texto, importando 367
arquivo, log de atuliazação 373
arquivos de configuração 253
arquivos de log 377
arquivos de log, nomes 276
arquivos de script 193
arquivos log 98
Arquivos Log 372
arquivos `my.cnf` 389
arquivos `tmp` 116
arquivos, `config.cache` 103
arquivos, limites de tamanho 10
arquivos, log binario 374
arquivos, log de consultas 373
arquivos, log de consultas lentas 377
arquivos, mensagem de erros 327
arquivos, permissões 924
arquivos, reparando 284
arredondamento, erros 529
arredondamento, erros de 483
árvores fonte de desenvolvimento 100
aspas 470
aspas, em string 470
atualização, log 373
atualizando 120
atualizando da 4.0 para 4.1 121
atualizando da versão 3.20 para 3.21 129
atualizando da versão 3.21 para 3.22 129
atualizando da versão 3.22 para 3.23 127
atualizando para uma arquitetura diferente 131
atualizando tabelas 46
atualizando, 3.23 para 4.0 123
atualizando, releases do MySQL 84
atualizando, tabela de permissões 130
aumentando a performance 412
aumentando a velocidade 378
AUTO-INCREMENT, ODBC 879
AUTO_INCREMENT 201
`AUTO_INCREMENT` e valores `NULL` 933

B

backup de banco de dados ... 361, 366
backups ... 275
backups, banco de dados ... 277
banco de dados, backups ... 275
banco de dados, criando ... 172
banco de dados, deletando ... 597
banco de dados, descarregando ... 361, 366
banco de dados, exibindo ... 369
banco de dados, informações sobre ... 192
banco de dados, links simbólicos ... 466
banco de dados, projetos ... 446
banco de dados, replicando ... 378
banco de dados, selecionando ... 173
banco de dados, usando ... 172
bancos de dados relacionais, definição ... 4
bancos de dados, nomes ... 472
barra invertida, caracter de escape ... 469
batch, modo ... 193
`batch`, `mysql` option ... 347
`BDB`, tabelas ... 46
`BDB`, tipo de tabela ... 630
benchmark, pacote ... 421
benchmarks ... 423
`BerkeleyDB`, tipo de tabela ... 630
biblioteca do servidor embutido MySQL ... 864
biblioteca `mysqlclient` ... 775
bibliotecas, lista de ... 953
Big5 Chinese, codificação de caracteres ... 930
binario, log ... 374
BitKeeper, árvores ... 100
`BLOB` , valores padrões em campos ... 498
`BLOB`, indexando colunas ... 603
`BLOB`, inserindo dados binários ... 470
`BLOB`, tamanho ... 503
`blocking_queries`, opção do `mysqlcc` option ... 355
Borland Builder 4 ... 876
Borland C++, compilador ... 888
buffer de cliente, tamanho ... 775
bugs, banco de dados ... 36
bugs, conhecidos ... 54
bugs, relatando ... 36
bugs.mysql.com ... 36

C

C API, tipos de dados ... 775
C++ ... 940
C++ Builder ... 878
C++, APIs ... 887
C++, compilador `gcc` ... 98
C, funções de instruções preparadas da API ... 831
cache de tabelas ... 451
caches, limpando ... 299
cálculo de datas ... 182
campos, alterando ... 610
campos, tipos ... 482
caracteres de escape ... 469
caracteres multi-byte ... 330
caracteres, conjunto de ... 99, 325
carregando tabelas ... 176
case sensitivo, em verificação de acesso ... 234
caso sensitivo nos nomes ... 473
caso sensitivo, de nomes de bancos de dados ... 44
caso sensitivo, de nomes de tabelas ... 44
caso-sensitivito, em pesquisas ... 930
caso-sensitivo, em comparação de string ... 519
cast ... 521
casts ... 504
certificação ... 14
ChangeLog ... 957
changes, version 3.19 ... 1076
changes, version 3.20 ... 1069
changes, version 3.21 ... 1056
changes, version 3.22 ... 1042
changes, version 3.23 ... 1000
`character-sets-dir`, opção `mysql` ... 347
chaves ... 449
chaves estrangeiras ... 50, 198, 611
chaves multi-column ... 450
chaves primárias, deletando ... 610
chaves, pesquisando em duas ... 200
cheio, disco ... 928
Chinese ... 930
citando dados binários ... 470
cliente, depurando ... 1085
clientes em threads ... 863
clientes, construindo clientes ... 862
coerção, operadores ... 521
colchetes ... 482
ColdFusion ... 876
colunas alterando ... 610
colunas, alterando ... 938
colunas, exibindo ... 369
colunas, exigências de armazenamento ... 502
colunas, índices ... 449
colunas, nomes ... 472
colunas, outros tipos ... 502
colunas, selecionado ... 179
comandos fora de sincronia ... 920
comandos SQL, replicação do master ... 400
comandos SQL, replicação do slave ... 402
comandos, lista de ... 350
comandos, para a distribuição binária ... 91
comandos, replicação do master ... 400
comandos, replicação do slave ... 402
comandos, sintaxe ... 3
combinação de padrões ... 185
comentários de colunas ... 602
comentários, adicionando ... 479
comentários, iniciando ... 51
comercial, tipos de suporte ... 17
compactadas, tabelas ... 336, 635
companias colaboradoras, lista de ... 955
comparação de string, case-sensitivo ... 519
Compatibilidade com o Ano 2000 ... 11

compatibilidade com ODBC 505, 506
compatibilidade com Oracle 614
Compatibilidade com Oracle 45
compatibilidade com Sybase 614
compatibilidade entre as versões do MySQL .. 121, 123
compatibilidade entre versões do MySQL 127, 129
compatibilidade padrões 42
compatibilidade, com ODBC ... 473, 483, 568, 601
compatibilidade, com Oracle 45, 558
compatibilidade, com padrão SQL 42
compatibilidade, com PostgreSQL 45
compatibilidade, with mSQL 520
compatibilidade, Y2K 11
compatibility, with ODBC 1063
compilação, otimizando 453
compilação, velocidade 456
compilador C++ não pode criar executáveis ... 104
compilador, C++ `gcc` 98
compilando funções definidas por usuários 906
compilando no Windows 134
compilando, estaticamente 98
compilando, problemas 103
`compress`, opção do `mysqlcc` 355
`compress`, opção `mysql` 347
concedendo privlégios 254
Conditions 768
conectando ao servidor 168, 238
conectando remotamente com SSH 133
conectando, verificação 239
Conector/ODBC 870
conexão abortada 918
confgurando senhas 266
config.cache 103
configuração pós-instalação 111
configuração, opções de 98
`configure`, executando depois da invocação anterior 103
`configure`, script 98
Conjunto de caracteres 710
conjunto de caracteres, adicionando 327
`connect_timeout` variable 356
`connect_timeout`, variável 350
`connection_name`, opção do `mysqlcc` 355
Connector/J 880
constante, tabela 425, 432
construindo programs clientes 862
consultas lentas, log 377
consultas, estimando performance 431
consultas, exemplos 195
consultas, fazendo 169
consultas, log 373
consultas, projeto de Estudo de Gêmeos 202
consultas, velocidade 423
consultoria 14
contando registros em uma tabela 188
contatos, informação 16
contribição, programas 940
controle de acesso 239
convenções tipográficas 2
convenções, tipográficas 2
conversores 942
copiando tabelas 600
copyrights 17
crackers, segurança contra 229
crash 1079
crash-me 422
crash-me, programa 419, 421
criando banco de dados 172
criando opções de inicialização padrão 216
criando tabelas 174
criando, relatórios de bug 36
criptografia de senha, reversibilidade 547
Cursors 770
custos de suporte 17
cvs, árvore 100

D

dados, carregando em tabelas 176
dados, conjunto de caracteres 325
dados, importando 367
dados, recuperando 177
dados, tamanho 446
Data e Hora, Tipos 489
data, tipos 502
`database`, opção do `mysqlcc` 355
`database`, opção `mysql` 347
DataJunction 876
datas, calculando 182
date values, problems 495
DATE, problemas com coluna 931
`DBI` Perl, modulo 881
`DBI`, interface 881
`DBI/DBD` 887
DBUG, pacote 1085
`debug`, opção `mysql` 347
`debug-info`, opção `mysql` 349
`default-character-set`, opção `mysql` 347
definição, bancos de dados 4
delayed_insert_limit 583
deleção, `mysql.sock` 929
deletando chaves primárias 610
deletando funções 900
deletando índices 610, 614
deletando linhas 933
deletando tabelas 612
deletando usuário 264
deletando usuários 264
deletando, banco de dados 597
Delphi 877, 941
depurando o cliente 1085
depurando o servidor 1079
derived tables 575
desatualizando 120

descarregando banco de dados 361, 366
descarregando tabelas 177
desconectando do servidor 168
desenvolvedores, lista de 945
desligando o server 113, 114
dicas, otimização 440
digitos 482
dinâmicas, tabelas 635
direito de escrita no tmp 116
disco cheio 928
disco, detalhes 464
discos, dividindo dados entre 468
display, tamanho 482
distribuição binária do MySQL 80
distribuição binária, instalando 91
distribuição binaria, on HP-UX 153
distribuição fontes do MySQL 80
distribuições binárias 86
distribuições binárias no Linux 141
distribuições fontes, instalando 94
DNS 459
download, fazendo 75

E

Eiffel Wrapper 888
emprego com MySQL 16
emprego, informação para contatos 16
`enable-named-commands`, opção `mysql` 348
endereço da mailing list 2
endereço eletrônico, para suporte à clientes 41
entre aspas, string 884
ENUM, tamanho 503
Errcode 371
errno 371
erro de aceeso negado 912
erros comuns 911
erros conhecidos 54
erros de arredondamento; 529
erros de ligação 922
erros internos de compilação 103
erros, checksum error 144
erros, conhecidos 54
erros, directory checksum error 144
erros, linguagem 327
erros, lista de 912
erros, relatando 2, 36
erros, relatórios 33
erros, tartamento para UDFs 906
erros, verificando tabelas 288
escolhendo tipos 501
escolhendo, uma versão do MySQL 80
espaço de armazenamento, minimizando 446
Espaço, Extençao no MySQL 733
estabilidade 8
estaticamente, compilando 98
estimando performance de consultas 431
estrangeiras, chaves 50, 198
estrutura de diretório, padrão 83
Estudos de Gêmeos, consultas 202
etiqueta para a rede 35, 41
Excel 876
executando `configure` depois da invocação prioritária 103
executando consultas 169
executando múltiplos servidores 219
executando um servidor web 20
executando, modo ANSI 42
executando, modo batch 193
`execute`, opção `mysql` 347
exemplos de consultas 195
exemplos de saida do `myisamchk` 293
exemplos, tabelas compactadas 338
exibindo informação de banco de dados 369
expressão, aliases 562
expressões extendidas 185
expressões regulares, sintaxe 1093
expressões, aliases 563
extensões ao, padrão SQL 42
extraindo datas 182

F

falhas repetidas 925
falhas, recuperação 287
`fatal signal 11` 103
fazendo consultas 169
fazendo parceria com a MySQL AB 15
fechando tabelas 451
ferramenta de clientes 775
ferramenta de linha de comando 347
ferramentas, lista de 955
ferramentas, mysqld_multi 333
ferramentas, mysqld_safe 331
ferramentas, safe_mysqld 331
flush tables 357
`force`, opção `mysql` 348
foreign key, restrição 52
foreign keys 50
FreeBSD, resolvendo problemas 105
full-text, pesquisa 619
FULLTEXT 619
funções binárias, exemplos 201
funções de data, compatibilidade Y2K 11
funções definidas por usuário, adicionando 899
funções definidas por usuários, adicionado 900
funções nativas, adicionando 908
funções para cláusulas `SELECT` e `WHERE` 504
funções, agrupando 504
funções, API C 778
funções, deletando 900
funções, instruções preparadas da API C 831
funções, novas 899
fuso horário, problemas 930

G

`gcc` ... 98
gdb, usando ... 1081
General Public License ... 5
General Public License, MySQL ... 18
geoespacial, recurso ... 733
geográficos, reucursos ... 733
geometria ... 733
GIS ... 733
globais, privilégios ... 254
GPL, Licença Publica Geral ... 1096
GPL, Licença Pública Geral do GNU ... 1096
GPL, MySQL ... 18
graphical tool ... 354
gratis, licensiamento ... 19
`GROUP BY`, aliases em ... 562
`GROUP BY`, extensões ao padrão SQL ... 565
`GROUP BY`, extensões para o padrão SQL ... 561
GUI tool ... 354

H

Handlers ... 769
`HEAP`, tipo de tabela ... 630
`help`, opção do `mysqlcc` ... 355
`help`, opção `mysql` ... 347
hints ... 43, 565
história do MySQL ... 5
histórico, arquivo de ... 345
`history_size`, opção do `mysqlcc` ... 355
hora, tipos ... 502
`host`, opção do `mysql` ... 355
`host`, opção `mysql` ... 348
`host`, tabelas ... 244
HP-UX, distribuição binaria ... 153
`html`, opção `mysql` ... 348

I

ID único ... 862
idade, calculando ... 182
`ignore-space`, `mysql` option ... 348
importando dados ... 367
índices ... 613
indices e colunas `BLOB` ... 603
índices e colunas `TEXT` ... 603
índices e `IS NULL` ... 449
índices e valores `NULL` ... 603
índices multi-coluna ... 450
índices multi-parte ... 613
índices, colunas ... 449
índices, deletando ... 610, 614
indices, e `LIKE` ... 448
índices, nomes ... 472
índices, prefixo mais a esquerda de ... 448
índices, tamanho de blocos ... 314
índices, uso de ... 447
informações gerais ... 1
inicialização, parâmetros ... 454
inicializando o servidor automaticamente ... 118
iniciando o `mysqld` ... 923
iniciando o servidor ... 111
iniciando vários servidores ... 219
iniciando, comentários ... 51
`InnoDB`, tabelas ... 46
`InnoDB`, tipo de tabela ... 630
inserção, velocidade ... 438
INSERT DELAYED ... 582
instalação de pacotes, Mac OS X PKG ... 71
instalação, layouts de ... 83
instalação, visão geral ... 60
instalando o Perl ... 164
instalando, distribuição binária ... 91
instalando, distribuições fontes ... 94
instalando, pacotes RPM do Linux ... 69
instalando, Perl no Windows ... 165
installing, funções definidas por usuários ... 906
inteiros ... 471
internals ... 896
Internet Relay Chat ... 41
Internet Service Providers ... 19
interno, travamento ... 443
IRC ... 41
`ISAM`, tipo de tabela ... 630
ISP serviços ... 19

J

Java, conectividade ... 880
JDBC ... 880

K

key space, MyISAM ... 633
keys, foreign ... 50

L

layouts de instalação ... 83
libmysqld ... 864
Licença Pública Geral (GPL) ... 5
licenças ... 17
licenciamento, custo ... 17
licenciamento, exemplos ... 18
licenciamento, gratis ... 19
licenciamento, informação para contatos ... 16
licenciamento, políticas ... 18
licenciamento, termos ... 17
ligação ... 862
ligação, erros ... 922
ligação, velocidade ... 456
limitações de projeto ... 418
limites, tamanho de arquivo ... 10
limpando caches ... 299
lingação, problemas com ... 862
línguas, suporte ... 327

linha de comando, ferramentas ... 347
linha de comando, histórico ... 345
linha de comando, opção `mysql` ... 347
linha de comando, opções ... 207
linha de comando, opções, `mysqlcc` ... 355
linhas, deletando ... 933
links simbólicos ... 465
links symbolicos ... 468
Linux, distribuição binárias ... 141
lista de colaboradores ... 948
lista de discussão MySQL ... 33
listas de discussão ... 33
listas de email ... 33
listas de mensagens, guias ... 41
listas de mensagens, localização dos arquivos ... 35
literais ... 469
`local-infile` ... 350, 356
localização padrão da instalação ... 83
lock de tabelas ... 443
lock de, registro ... 49
locks ... 453
log, alterações ... 957
log, arquivos ... 98, 377
logomarcas ... 20

M

Mac OS X, instalação ... 71
`make_binary_distribution` ... 331
mantendo, arquivos de log ... 377
manual, convenções tipográficas ... 2
manual, formatos disponíveis ... 2
manual, online location ... 2
manutenção, tabelas ... 292
marcas registradas ... 20
master/slave, configuração ... 379
max memory used ... 358
`max_allowed_packet` ... 350, 356
`max_join_size` ... 350, 356
MBR ... 756
memória virtual, problemas quando compilando ... 103
memoria, uso ... 457
memory use ... 357
mensagem de erro, exibindo ... 371
mensagem de erro, linguagem ... 327
mensagens de erro, arquivo não encontrado ... 924
MERGE, definição de tabelas ... 638
`MERGE`, tipo de tabela ... 630
meta caracteres, e `LIKE` ... 448
meta caracteres, na tabela `mysql.columns_priv` ... 243
meta caracteres, na tabela `mysql.db` ... 243
meta caracteres, na tabela `mysql.host` ... 243
meta caracteres, na tabela `mysql.tables_priv` ... 243
metacaracteres, na tabela `mysql.user` ... 239
métodos, travamento ... 1087
Minimum Bounding Rectangle ... 756
MIT-pthreads ... 106
modo ANSI, executando ... 42
modo batch ... 193
módulos, lista dos ... 9
monitor, terminal ... 168
mostrando informações de banco de dados ... 369
mostrando, informações, `SHOW` ... 303
mostrando, status da tabela ... 304
mSQL compatibilidade ... 520
`msql2mysql` ... 346
mudança de privilégios ... 245
multi mysqld ... 333
multi-byte, caracteres ... 330
multi-byte, conjunto de caracteres ... 921
multi-colunas, índices ... 450
multi-parte, índice ... 613
múltiplos servidores ... 219
My, derivação ... 5
`my.cnf`, arquivo ... 389
MyISAM, tabelas compactadas ... 336, 635
MyISAM, tamanho ... 503
`MyISAM`, tipo de tabela ... 630
`myisamchk` ... 100, 331
`myisamchk`, exemplos de saída ... 293
`myisamchk`, opções ... 282
`myisampack` ... 336, 608, 635
MyODBC ... 870
MyODBC, relatando problemas ... 879
`mysladmin` ... 356
`mysql` ... 346, 347
MySQL AB, definição ... 12
MySQL, certificação ... 14
MySQL, consultoria ... 14
MySQL, definições ... 4
MySQL, história do ... 5
MySQL, introdução ... 4
`mysql`, opção de linha de comando ... 347
MySQL, pronúncia ... 5
MySQL, tipo de tabelas ... 630
MySQL, treinamento ... 14
MYSQL, versão do ... 75
mysql.sock, alterando localização do ... 98
`mysql.sock`, proteção ... 929
`mysql_fix_privilege_tables` ... 250
`mysql_install_db` ... 331
`mysql_install_db`, script ... 115
`mysqlaccess` ... 346
`mysqladmin` ... 299, 301, 305, 346, 597, 598
`mysqlbinlog` ... 346, 358
`mysqlbug` ... 331
`mysqlcc` ... 346, 354
`mysqlcc`, opções de linha de comando ... 355
`mysqlclient`, biblioteca ... 775
`mysqld` ... 331
`mysqld` , tamanho de buffer do servidor ... 454
`mysqld`, iniciando ... 923
`mysqld`, opções ... 454

`mysqld-max` ... 343
`mysqld_multi` ... 333
`mysqld_safe` ... 331
`mysqldump` ... 132, 346, 361
`mysqlimport` ... 132, 346, 367, 589
`mysqlshow` ... 346
mysqltest, Programa de Tste do MySQL ... 896

N

named pipes ... 68
não delimitadas, strings ... 494
não encontrados, regitros ... 934
não pode criar/gravar arquivos ... 919
não transacionais, tabelas ... 916
nativa, suporte de thread ... 78
nativas, adicionando funções ... 908
`net_buffer_length` ... 350, 356
NetWare ... 74, 164
`no-auto-rehash`, opção `mysql` ... 347
`no-beep`, `mysql` option ... 347
`no-named-commands`, opção `mysql` ... 348
`no-pager`, opção `mysql` ... 348
`no-tee`, opção `mysql` ... 348
nome de máquina, armazenando em cache ... 459
nome de maquinas padrao ... 238
nome de usuários e senhas ... 259
Nome do Golfinho do MySQL ... 5
Nome do MySQL ... 5
nomeando, releases do MySQL ... 81
nomes ... 472
nomes alias, caso sensitivo ... 473
nomes de banco de dados, caso sensitivo ... 44
nomes de bancos de dados, caso sensitivo ... 473
nomes de colunas, caso sensitivo ... 473
nomes de tabelas, caso sensitivo ... 44, 473
nomes validos ... 472
nomes, caso sensitivo ... 473
nomes, variáveis ... 474
NOT NULL, restrições ... 53
notação de máscara de rede, na tabela `mysql.user` ... 240
Novell NetWare ... 74, 164
novos procedimentos, adcionado ... 909
novos usuários, adicionando ... 92
`NULL` e colunas `AUTO_INCREMENT` ... 933
`NULL` e colunas `TIMESTAMP` ... 933
`NULL` vs. valores vazios ... 932
`NULL`, teste para nulo ... 508, 511
`NULL`, testes para nulo ... 506
NULL, valores ... 185
`NULL`, valores e índices ... 603
numericos, tipos ... 502
número de ponto flutuante ... 483
números ... 471
números de release ... 80
números válidos, exemplos ... 471

O

O que é criptografia? ... 268
O que é um X509/Certificado? ... 268
objetivos do MySQL ... 5
obtendo o MySQL ... 75
ODBC ... 870
ODBC compatibilidade ... 568
ODBC compatibility ... 1063
ODBC, administrador ... 871
ODBC, compatibilidade ... 473, 483, 601
ODBC, compatibilidade com ... 505, 506
odbcadmin program ... 877
OLEDB ... 940
`one-database`, `mysql` option ... 348
online location of manual ... 2
opção config-file ... 333
opção configure, –with-low-memory ... 103
opção de linha de comando `mysql` ... 347
opção example ... 334
opção help ... 334
opção log ... 334
opção mysqladmin ... 334
opção mysqld ... 334
opção no-log ... 334
opção password ... 334
opção tcp-ip ... 334
opção user ... 334
opção version ... 334
opçãoes de inicialização, padrão ... 216
opções de linha de comando, `mysqlcc` ... 355
opções de reparo, myisamchk ... 284
opções de verificaçã, myisamchk ... 283
opçoes frornecidas pelo MySQL ... 168
opções mysqld ... 207
opções padrões ... 216
opções, configuração ... 98
opções, `myisamchk` ... 282
opções, replicação ... 389
`Open Source`, definição ... 5
open tables ... 357
OpenGIS ... 733
opens ... 357
OpenSSL ... 268
operações aritiméticas ... 522
operadores de coerção ... 521, 522
optimizações ... 432
optimizando, tabelas ... 291
Oracle, compatibilidade ... 558, 614
ordenação, conjunto de caracteres ... 325
ordenando dados ... 180
ordenando registros de tabelas ... 180
ordenando strings ... 329
ordenando tabela de permissões ... 241, 243
`ORDER BY`, aliases em ... 562
otimização, dicas ... 440
otimizando DISTINCT ... 434
otimizando, LEFT JOIN ... 435
otimizando, LIMIT ... 437

P

`pack_isam` ... 336
pacotes, lista de ... 954
padrao do nome de maquinas ... 238
padrão, privilégios ... 260
Padrãos SQL, diferenças do ... 258
padrões, combinação ... 185
padrões, embutico ... 866
`pager`, opção `mysql` ... 348
palavras reservadas, exceções de ... 479
palavras-chave ... 479
par6ametros de inicialização, `mysqlcc` ... 355
paramentros de inicialização, sintonia ... 453
parâmetros de inicialiazação, `mysql` ... 347
parâmetros, servidor ... 454
parando o servidor ... 118
`password`, opção do `mysqlcc` ... 355
`password`, opção `mysql` ... 349
patches, aplicando ... 97
performance, aumentando ... 412, 446
performance, benchmarks ... 423
performance, detalhes de disco ... 464
performance, estimando ... 431
Perl API ... 881
Perl DBI/DBD, problemas de instalação ... 166
Perl, instalando ... 164
Perl, instalando no Windows ... 165
Perl, modulos ... 940
permissões de tabelas ... 245
perror ... 371
pesquisa full-text ... 619
pesquisando em duas chaves ... 200
pesquisas e caso-sensitivito ... 930
PHP API ... 880
`plugins_path`, opção do `mysqlcc` ... 355
ponto decimal ... 482
ponto flutuante ... 471
`port`, opção do `mysqlcc` ... 355
`port`, opção `mysql` ... 349
portabilidade ... 419
portabilidade, tipos ... 502
portando para outros sistemas ... 1078
pós-instalação, configuração e testes ... 111
pós-instalação, multiplos servidores ... 219
PostgreSQL, compatibilidade ... 45
preço do suporte ... 17
prefixo mais a esquerda de índices ... 448
PRIMARY KEY, restrição ... 52
privilégio de acesso ... 226
privilégio, mudança ... 245
privilégio, sistemas de ... 232
privilégios de usuário, apagando ... 264
privilégios de usuário, deletando ... 264
privilégios globais ... 254
privilégios, adicionando ... 261
privilégios, apagando ... 264
privilégios, concedendo ... 254
privilégios, deletando ... 264
privilégios, exibir ... 322
privilegios, localização de informações ... 236
privilégios, padrão ... 260
privilégios, revogando ... 254
problemas `cc1plus` ... 103
problemas com bloqueios de tabela ... 444
problemas com fuso horário ... 930
problemas de ligação ... 922
problemas e erros comuns ... 911
problemas inicializando o servidor ... 116
problemas instalando no Solaris ... 144
problemas, colunas `DATE` ... 931
problemas, compilando ... 103
problemas, erros de acesso negado ... 912
problemas, instalando no IBM-AIX ... 155
problemas, instalando Perl ... 166
problemas, ODBC ... 879
problemas, relatando ... 36
problems, date values ... 495
procedimentos, adicionando ... 909
procedures, stored ... 49, 763
processando, argumentos ... 904
processos, exibindo ... 320
processos, suporte de ... 78
procurando, páginas web MySQL ... 35
produtos, vendendo ... 18
Programa Access ... 874
programa, crash-me ... 419
programas clientes ... 862
programas, contribuição ... 940
programas, lista de ... 331, 345
projeto, limitações ... 418
projetos, detalhes ... 54
projetos, escolhas ... 446
`prompt` de comando ... 353
`prompt`, `mysql` option ... 347
prompts, significados ... 171
pronúncia, MySQL ... 5
Protocol mismatch ... 129
`protocol`, `mysql` option ... 349
`protocol`, `mysql`, opção ... 368
Protocolos não correspondetes ... 129
Provedores de Serviços de Internet ... 19
Python, API ... 888

Q

queries, exemplos ... 195
Query Cache ... 625
`query`, opção do `mysqlcc` ... 355
questions ... 357
questões, respondendo ... 41
`quick`, `mysql` option ... 349

R

RAID, erros de compilação ... 104
RAID, table type ... 606
`raw`, opção `mysql` ... 349
reconfigurando ... 103
`reconnect`, `mysql` option ... 349
recriando tabelas de permissões ... 261
recuperando dados de tabelas ... 177
recupração em caso de falhas ... 287
recursos do MySQL ... 6
recursos principais do MySQL ... 6
RedHat Package Manager ... 69
reduzindo o tamanho dos dados ... 446
referências ... 611
regex ... 1093
`register`, opção do `mysqlcc` ... 355
registro, lock de ... 49
registro, ordenando ... 180
registros não encontrados ... 934
registros, contando ... 188
registros, secionando ... 178
relatando problemas com MyODBC ... 879
relatando, bugs ... 36
relatando, erros ... 2
relatório de bug, critério para ... 38
relatório, erros ... 33
releases, atualizando ... 84
releases, esquema de nomenclatura ... 81
releases, testando ... 82
reordenando colunas ... 938
repairando tabelas ... 289
`replace` ... 346
replicação ... 378
replicação do master, comandos ... 400
replicação do slave, comandos ... 402
Resolvendo problemas no FreeBSD ... 105
Resolvendo problemas no Solaris ... 105
respondendo às questões, etiqueta ... 41
Responsáveis pela Documentação, lista dos ... 952
Retângulo de Limite Mínimo ... 756
revogando privlégios ... 254
root, recuperando senha do ... 927
root, senha ... 260
RTS-threads ... 1088

S

`safe-mode`, comando ... 351
`safe-updates`, opção `mysql` ... 349
`safe_mysqld` ... 331
Sakila ... 5
script `mysqlbug` ... 36
script `mysqlbug`, localização ... 2
script, arquivos de ... 193
scripts ... 331, 333, 347
scripts `mysql_install_db` ... 115
scripts, `mysqlbug` ... 36
segurança com transação, tabelas ... 643
segurança contra crackers ... 229
segurança de sistema ... 226
selecionando banco de dados ... 173
`SELECT`, Query Cache ... 625
`select_limit` ... 350, 356
senha, configuração ... 462
senha, usuário root ... 260
senhas esquecidas ... 927
senhas seguras ... 232
senhas, configurando ... 257, 266
senhas, para usuários ... 259
senhas, recuperando ... 927
sequência de chamadas para funções agregadas, UDF ... 903
sequência de chamadas para funções simples, UDF ... 902
sequencia de emulação ... 552
`server`, opção do `mysqlcc` ... 355
serviços, ISP ... 19
serviços, web ... 19
servidor web, executando ... 20
servidor, administração ... 356
servidor, conectando ... 168, 238
servidor, depurando ... 1079
servidor, desligar o ... 113, 114
servidor, disconectando ... 168
servidor, inicializando e parando ... 118
servidor, iniciando o ... 111
servidor, problemas inicializando o ... 116
servidores, múltiplos ... 219
SET, tamanho ... 503
`set-variable`, opção `mysql` ... 348
shell, sintaxe ... 3
`silent`, `mysql` option ... 349
simbólicos, links ... 465
sintaxe de expressões regulares, descrição ... 1093
sistema de privilégios ... 232
sistema de privilégios, dexcriçã ... 232
sistema, otimização ... 453
sistema, segurança ... 226
sistema, tabela de ... 425
sistema, variáveis ... 475
sistemas operacionais, limites de tamanho de arquivo ... 10
sistemas operacionais, suportados ... 78
sistemas operacionais, Windows versus Unix ... 134
sites de espelhos ... 75
`skip-column-names`, opção `mysql` ... 348
`skip-line-numbers`, `mysql` option ... 348
slow queries ... 357
socket, alteração localização dos ... 98
`socket`, opção do `mysqlcc` ... 355
`socket`, opção `mysql` ... 349
Solaris, problemas de instalação ... 144
Solaris, resolvendo problemas ... 105
SQL, definição ... 4
SQL-92, extensões ao ... 42
`sql_yacc.cc` problemas ... 103

SSH 133
SSL e X509 Basicos 268
SSL, opções de linha de comando 274
SSL, opções relacionadas 273
`status`, comando 351
status, resultado do comando 357
status, tabelas 304
stored procedures 763
stored procedures e triggers, definição 49
string entre aspas 884
string tipos 496
string, ordenação 329
string, tipos 503
strings não delimitadas 494
strings, caracteres de escape 469
strings, definição 469
striping, definição 464
subqueries 570
subquery 570
subselects 570
sugestões 568
superusuário 260
suporte a clientes, endereço eletrônico 41
suporte à thread não-nativa 106
suporte técnico, endereço eletrônico 41
suporte técnico, licenciamento 18
suporte, custo 17
suporte, endereço eletrônico 41
suporte, licenciamento 18
suporte, para sistemas operacionais 78
suporte, termos 17
suporte, tipos 17
supressão, valores padrão 99
supressão, valores padrões 53
Sybase, compatibilidade 614
`syntax`, opção do `mysqlcc` 355
`syntax_file`, opção do `mysqlcc` 355

T

tabela `db`, ordenando 243
tabela de permissões, ordenando 241, 243
tabela de sistema 425
tabela está cheia 919
tabela `host` 244
tabela `host` table, ordenando 243
tabela `user`, ordenando 241
tabela, alterando 610
tabela, aumnetando performance 446
tabela, permissões 245
tabelas abertas 451
tabelas `BDB` 698
tabelas `Berkeley DB` 698
tabelas compactadas 336
tabelas compactadas, formato 635
tabelas constantes 432
tabelas de permissões, recriando 261
tabelas dinâmicas, características 635
tabelas `HEAP` 642
tabelas ISAM 641
tabelas temporárias, problemas 939
tabelas transacionais 46
tabelas, abrindo 451
tabelas, aliases 564
tabelas, alterando 938
tabelas, alterando a ordem das colunas 938
tabelas, atualizando 46
tabelas, carregando dados 176
tabelas, constante 425
tabelas, contando regitros 188
tabelas, copiando 600
tabelas, criando 174
tabelas, deletando 612
tabelas, deletando linhas 933
tabelas, descarregando 361, 366
tabelas, desfragmentação 635
tabelas, desfragmentar 292
tabelas, exibindo 369
tabelas, fechando 451
tabelas, fusão de 638
tabelas, ID único para o último registro 862
tabelas, informações 293
tabelas, informações sobre 192
tabelas, laterando 608
tabelas, mostrando status 304
tabelas, muitas 453
tabelas, nomes 472
tabelas, optimizando 291
tabelas, ordenando registros 180
tabelas, recuperando dados 177
tabelas, regime de manutenção 292
tabelas, reparando 289
tabelas, selecinando colunas 179
tabelas, selecionado registros 178
tabelas, tamanho máximo 10
tabelas, tipos 630
tabelas, travamento 443
tabelas, verificação 283
tabelas, verificando erros 288
table is full 461
`table`, opção `mysql` 349
tables, desfragmentação 298
tables, flush 357
tables, fragmentação 298
tables, multiple 190
tables, RAID 606
tamanho das tabelas 10
tamanho de buffer do servidor `mysqld` 454
tamanho, display 482
tar, problemas no Solaris 144
Tcl, API 888
TCP/IP 68
tebela de permissões, atualizando 130
`tee`, opção `mysql` 349
tempo esgotado 313
tempo excedido, `connect_timeout` variável 350

tempo limite 553, 583
terminal monitor, definição 168
testando a instalação 112
testando mysqld, mysqltest 896
testando o servidor 112
testando, conectando ao servidor 239
testando, releases do MySQL 82
testes pós-instalação 111
testing, installation 112
Texinfo 2
`TEXT` , valores padrões em campos 498
`TEXT`, indexando colunas 603
`TEXT`, tamanho 503
texto, arquivo 367
thread, suporte de 78
threads 357, 896
threads, clientes em 863
threads, diferenças entre pacotes 1090
threads, exibindo 320
threads, RTS 1088
timeout 313
timeout, `connect_timeout` variable 356
`TIMESTAMP` e valores `NULL` 933
tipo, conversão de 504
tipos de campos 501
tipos de dados, C API 775
Tipos de data e hora 489
tipos de data, Y2K assuntos 490
tipos de suporte 17
tipos de tabela, escolhedo 630
tipos string 503
tipos strings 496
tipos, campos 482
tipos, data 502
tipos, hora 502
tipos, numéricos 502
tipos, portabilidade 502
ToDo list for MySQL 27
TODO, servidor embutido 866
TODO, symlinks 467
tools, graphical 354
tools, GUI 354
trabalhos no MySQL 16
Tradutores, lista de 952
transações, suporte 46, 643
`translations_path`, opção do `mysqlcc` 355
tratando erros 906
travamento, métodos 1087
treinamento 14
triggers 49
tutorial 168

U

UCS-2 710
UDFs, compilando 906
UDFs, definição 899
UDFs, valor de retorno 906
último registro, ID único 862
`unbuffered`, opção `mysql` 348
único, ID 862
Unicode 710
UNIQUE, restrição 52
unnamed views 575
uptime 357
URLS para download do MySQL 75
usando multiplos discos para guardar dados 468
`user`, opção do `mysqlcc` 356
`user`, opção `mysql` 349
uso de memória, myisamchk 286
uso, do MySQL 420
usuário, funções de finidas, adicionando 900
usuários, adicionando 92
usuários, adicionando privilégios 261
usuários, deletando 264
usuários, do MySQL 420
usuários, funções definidas por 899
usuarios, root 260
usuários, variáveis 474
UTF-8 710
utilitários 944

V

valor de retorno, UDFs 906
valores negativos 471
valores padrão, supressão 99
valores padrões 418, 579, 602
valores padrões em campos `BLOB` e `TEXT` 498
valores padrões, supressão 53
varáveis de ambiente 216
`VARCHAR`, tamanho 503
variáveis de ambiente 253, 331, 345
variáveis de ambientes, lista 1092
variáveis de sistema 475
variáveis de usuários 474
variaveis, `mysqld` 454
variáveis, status 306
variáveis, valores 312
velocidade da inserção 438
velocidade das consultas 423, 431
velocidade de compilação 456
velocidade de ligação 456
velocidade, aumentado 378
vendendo produtos 18
`verbose`, opção `mysql` 349
verificação de permissões, efeito na velocidade 423
verificando erros em tabelas 288
versão, escolhendo 80
versão, última 75
`version`, opção do `mysqlcc` 356
`version`, opção `mysql` 349
`vertical`, opção `mysql` 347
views 51
visão geral 1

visão geral da instalação 94
Visual Basic 878

W

`wait`, opção `mysql` 349
Well-Known Binary, formato 741
Well-Known Text, formato 740
Windows 870
Windows, assuntos em aberto 136
Windows, atualizando 132
Windows, compilando no 134
Windows, versus Unix 134
WKB 741
WKT 740
Word 877
wrappers, Eiffel 888

X

`xml`, `mysql` option 348